

الدكتور مادلين عبود

الدكتور مراكا نمرزوق

قواعد المعطيات (1)

الطبعة الأولى

ق المؤلف والطبع والنشر محفوظة لجامعة دمشق

١٤٢١ - ١٤٢٢ هـ
٢٠٠٠ - ٢٠٠١ م

جامعة دمشق



جامعة دمشق
كلية المعلوماتية

قواعد المعطيات (1)

الطبعة الأولى

الدكتور مراكا نمرزوق

الدكتورة مادلين عبود

2000 - 2001

منشورات جامعة دمشق

مقدمة الكتاب

يُجمع العاملون في المعلوماتية على اعتبار قواعد المعطيات أحد أوسع علوم المعلوماتية استثماراً وأكثرها فائدة في الحياة العملية لمهندسي المعلوماتية. ويتأكد ذلك من خلال ملاحظة النسبة المرتفعة من التطبيقات المعلوماتية، التي تؤول إلى تصميم وتطوير قاعدة معطيات، تتضمن معلومات المؤسسة التي تسعى لاستخدام الوسائل المعلوماتية في أعمالها، والبرامج التي تحقق الوظائف المتوقعة من النظام المعلوماتي.

لقد تولد لدى أوائل العاملين في التطبيقات الإدارية والمالية للمعلوماتية، العديد من الأفكار التي تجلت في البداية كمنهجيات ومبادئ عامة، تفيد في جعل العمل البرمجي يتحول من عمل يعتمد إلى حد بعيد على المهارات الفردية للمبرمجين وفهمهم للمسألة المطروحة، إلى عمل هندسي يمتلك معايير ومنهجيات تحاكي ما توفره المهن الهندسية المرفقة كالهندسة المعمارية أو الهندسة الكهربائية.

وقد كان من أهم هذه الأفكار عزل المعطيات عن البرامج التي تعالجها وعن بنى ووسائط التخزين التي تسجل عليها.

ولاحظ العاملون في المعلوماتية تكراراً في الوظائف البرمجية التي يجدون أنفسهم مضطرين إلى إعادة كتابتها واختبارها في كل التطبيقات التي ينفذونها، فوجدوا أنه لا بد من تطوير نظام برمجي عام يعنى بتحقيق مجموعة من الوظائف العامة التي تفيد في تعريف بنى المعطيات وتبسيط التعامل مع المعطيات على مستوى عالٍ من التجريد. ومن هنا تولدت فكرة إنشاء أنظمة إدارة قواعد المعطيات.

لقد ساهم العاملون في قواعد المعطيات في إرساء عدد من المبادئ والمفاهيم التي جرى تطويرها في مراحل لاحقة لتكوين اللبنة الأولى في علم هندسة البرمجيات. واستفاد العاملون في تطوير أنظمة إدارة قواعد المعطيات من منهجيات وأدوات البرمجة التي أنتجها علم هندسة البرمجيات، وضمنوا جزءاً كبيراً منها في منتجاتهم البرمجية التي كرسوها لإدارة قواعد المعطيات.

لقد كان عقد السبعينيات حافلاً بالعديد من الأفكار والنظريات التي مهدت لظهور أنظمة قواعد المعطيات في بداية الثمانينيات. ومنذ ذلك الوقت ما نزال هذا النوع من الأنظمة يزداد تطوراً وترداداً بذلك أهميته ويتعمق دوره كأداة لا بد من إتقان استخدامها لتطوير الأنظمة المعلوماتية.

وساهم التنافس التجاري بين الشركات التي عملت على تطوير وتسويق أنظمة إدارة قواعد المعطيات في إغناء هذه الأنظمة. فقد سعت كل منها إلى إضافة مكونات ووظائف جديدة لنظم إدارة قواعد المعطيات التي تنتجها لتجعلها متفوقة على سواها، وهذا ما جعل هذه الأنظمة تتحول تدريجياً إلى محيط تطوير متكامل يوفر كافة الوظائف التي يحتاج إليها العاملون في تطوير أنظمة المعلومات خلال معظم مراحل المشروع المعلوماتي، بل إن بعض هذه الشركات ضمنت منتجاتها أدوات مساعدة في هندسة البرمجيات.

وقد كان لهذا التنافس بعض الآثار السلبية التي تجلت في عدم التوافق بين الأنظمة المتعددة، وهذا ما استدعى بذل جهود كبيرة في بداية التسعينيات للاتفاق على صيغ معيارية، خاصة فيما يتعلق بلغات قواعد المعطيات العلائقية. كما اجتمعت هذه الشركات في نهاية التسعينيات لتضع في متناول مستخدمي أنظمتها أدوات للربط بين مختلف هذه الأنظمة.

يتناول هذا الكتاب مادة قواعد المعطيات من وجهة النظر الأكاديمية متجنباً إلى حد بعيد الخصوصيات التي تقدمها أنظمة إدارة قواعد المعطيات التجارية.

يتبع من الفصل الأول التعاريف الأساسية، وعرضاً عاماً ومبسّطاً لأهداف قواعد المعطيات، والوظائف التي تحققها أنظمة إدارة قواعد المعطيات. يركز هذا الفصل على مستويات التجريد المعتمدة في تصميم قواعد المعطيات ويعرض البنية العامة لأنظمة إدارة قواعد المعطيات باعتبارها نظاماً برمجياً يحقق طيفاً واسعاً من الوظائف.

يتناول الفصلان الثاني والثالث المستويين الأساسيين في تصميم قواعد المعطيات. ففي الفصل الثاني يجري عرض مراحل إنشاء المخطط المفاهيمي لقاعدة معطيات، ويركز على نموذج كيانات-ارتباطات باعتباره أنجح النماذج وأوسعها انتشاراً في الوقت الراهن. ويعرض الفصل الثالث النموذج العلائقي من خلال عرض البنى الأساسية لقواعد المعطيات العلائقية والعمليات الأساسية والموسعة التي يتيحها الجبر العلائقي.

يكمل الفصل الرابع ما يتضمنه الفصل الثالث، فيعرض لغة SQL باعتبارها تراكيباً من لغة الجبر العلائقي والحساب العلائقي. وقد اختيرت هذه اللغة لأنها أكثر لغات الاستعلام انتشاراً. يبين هذه الفصل التراكيب الأساسية للغة SQL خاصة تلك التي تسمح بتعريف بنية المعطيات وإضافة وتعديل المعطيات في قاعدة المعطيات والاستعلام ضمنها.

يتضمن الفصل الخامس عرضاً لشروط التكامل باعتبارها الطريقة الأساسية في قواعد المعطيات للتوثيق من تناسق المعطيات وسلامة عمليات التعديل التي تُجرى عليها من جهة، ولحماية قاعدة المعطيات من التخريب والإخلال بتناسقها وصحتها من جهة أخرى. يعرض هذا الفصل الأشكال المختلفة لشروط التكامل، وطريقة تحقيقها.

ونظراً لأهمية التصميم الجيد لقواعد المعطيات، والانتشار الواسع لقواعد المعطيات العلائقية، فقد كررنا الفصل السادس للأسس النظرية التي يمكن اعتمادها لدى تصميم قاعدة معطيات علائقية

بهدف إتقان ظاهرة تكرار المعطيات إلى الحد الأدنى، والتوثق من تمثيل العلاقات الموجودة بين الواصفات، ولتسهيل اختبار مدى تحقيق التعديلات التي تجري على قاعدة المعطيات لشروط التكامل المعرفة عليها.

يعرض الفصل السابع المخطط الداخلي لقاعدة المعطيات، فيذكر بنى تخزين المعطيات التي درسها الطالب في مادة الخوارزميات وبنى المعطيات في السنة الثانية، وبين طريقة توظيف هذه البنى في قواعد المعطيات.

يتضمن الفصلان الثامن والتاسع عرضاً لوظيفتين هامتين من الوظائف التي تستند إلى نظم إدارة قواعد المعطيات هما إدارة العمليات الشاملة وإدارة الأعطال.

إننا إذ نضع هذا الكتاب بين أيدي طلابنا الأعزاء، نتمنى أن يسهم عملنا المتواضع هذا في دعم مسيرة المعلوماتية في التعليم العالي، ونأمل من طلابنا وزملائنا ألا يخلوا علينا بملاحظاتهم التي سنوليها كل اهتمام وتقدير وسنستفيد منها في تطوير هذا الكتاب. والله من وراء القصد.

المؤلفان

د. مراكح مرزوق

د. مادلين عبود

مركز الدراسات والبحوث العلمية

المعهد العالي للعلوم التطبيقية والتكنولوجيا

الفهرس

11	الفصل الأول : مدخل إلى قواعد المعطيات
11	مقدمة
13	1- الغرض من نظم إدارة قواعد المعطيات
15	1-1 تكرار المعطيات وتضاربها
16	2-1 صعوبة الوصول إلى المعطيات
17	3-1 عزل المعطيات
17	4-1 تعارض في الوصول المتزامن
18	5-1 أمن المعطيات
18	6-1 تكامل المعطيات
19	2- الوظائف التي توفرها أنظمة إدارة قواعد المعطيات
19	1-2 مركزية المعلومات
19	2-2 استقلال المعطيات
20	3-2 معالجة المعطيات بواسطة لغات غير إجرائية
20	4-2 التسهيلات الخاصة بإدارة المعطيات
21	5-2 الوصول إلى المعطيات بفعالية
21	6-2 التحكم في تكرار المعطيات
21	7-2 تكامل المعطيات
22	8-2 تقسيم المعطيات
22	9-2 أمن المعطيات
22	3- تصميم قواعد المعطيات
26	4- نماذج المعطيات Data Models

- 27 ----- 1-4- النماذج المنطقية المعتمدة على الأغراض
- 29 ----- 2-4- النماذج المنطقية المعتمدة على التسجيلات
- 34 ----- 3-4- النماذج الفيزيائية
- 34 ----- 5- المخططات والحالات
- 35 ----- 6- استقلال المعطيات
- 35 ----- 1-6- استقلال المعطيات فيزيائياً
- 35 ----- 2-6- استقلال المعطيات منطقياً
- 36 ----- 7- لغات قواعد المعطيات
- 36 ----- 1-7- لغة تعريف المعطيات (DDL : Data Definition Language)
- 37 ----- 2-7- لغة التعامل مع المعطيات (DML : Data Manipulation Language)
- 38 ----- 8- البنية العامة لقاعدة المعطيات
- 39 ----- 1-8- الاستفسار
- 39 ----- 2-8- مدير قاعدة المعطيات Data base Manager
- 41 ----- 3-8- مشغل قاعدة المعطيات Data base Administrator
- 43 ----- 4-8- مستخدمو قاعدة المعطيات
- 44 ----- 5-8- مدير الملفات
- 44 ----- 6-8- معالج الاستفسار
- 45 ----- 7-8- المترجم الأولي لتعليمات لغة التعامل مع المعطيات DML pre-compiler
- 45 ----- 8-8- مترجم لغة تعريف المعطيات DDL compiler
- 46 ----- تمارين الفصل الأول

- الفصل الثاني : المخطط المفاهيمي لقاعدة المعطيات- نموذج كيانات-ارتباطات 47
- مقدمة 47
- 1- الكيانات وصفوف الكيانات 48
- 2- الارتباط وصفوف الارتباطات 50
- 3- الواصفات 53
- 4- تمثيل الشروط 54
- 5- المفاتيح 56
- 5-1- المفتاح الأعلى أو المفتاح الرئيسي 56
- 5-2- Primary key المفتاح الأولي 57
- 5-3- وصف علاقة الارتباط 57
- 5-4- المفتاح الأولي لعلاقة الارتباط 58
- 5-5- صفوف الكيانات الضعيفة 59
- 6- مخطط تمثيل كيان-ارتباط E-R Diagram 60
- 7- نموذج كيان-ارتباط موسع 64
- 7-1- علاقة التخصيص 64
- 7-2- علاقة التعميم 66
- 7-3- التجميع Aggregation 69
- 8- مراحل التصميم في نموذج كيانات-ارتباطات 71
- 9- تحويل مخطط E-R إلى جداول 72
- 9-1- تمثيل صفوف الكيانات كجدول 73

74	9-2- تمثيل صفوف الارتباطات كجدول
74	9-3- الجداول المكررة
74	9-4- تمثيل التعميم كجدول
77	تمارين الفصل الثاني
81	الفصل الثالث النموذج العلاقي
81	مقدمة
81	1- بنية قواعد المعطيات العلاقية
82	1-1- البنى الأساسية
84	1-2- مخطط قاعدة المعطيات
85	1-3- المفاتيح
86	1-4- لغات الاستعلام
86	2- الجبر العلاقي
87	2-1- العمليات الأساسية
96	3- لغة القضايا The tuple Relational Calculus
98	4- العمليات الموسعة للجبر العلاقي
98	4-1- الإسقاط المعمم
99	4-2- الدمج الخارجي
102	4-3- التوابيع التجميعية Aggregate functions
103	5- تعديل قاعدة المعطيات
103	5-1- الحذف Deletion
104	5-2- الإضافة Insertion
104	5-3- التعديل Updates

- 6- المنظار Views 105
- 7- الخلاصة 107
- تمارين الفصل الثالث 109
- الفصل الرابع : لغة SQL 111
- مقدمة 111
- لمحة تاريخية 111
- 1- تعريف بلغة SQL 112
- 2- لغة الاستعلام 113
- 1-2- البنية الأساسية للاستعلام في لغة SQL 113
- 2-2- إعادة التسمية 116
- 3-2- لعمليات على سلسلة الخارف 118
- 4-2- ترتيب الحدوديات الناتجة 119
- 5-2- العمليات على المجموعات 120
- 6-2- التوابع التجميعية 122
- 7-2- معالجة القيم غير المعلومة 123
- 8-2- تجزئة العلاقة 124
- 9-2- الاستفسارات الجزئية المضمنة 126
- 10-2- العلاقات المشتقة 129
- 3- المناظير 130
- 4- تعديل قاعدة المعطيات 130
- 1-4- الحذف 131

132	4-2- الإضافة
134	4-3- التعديل
135	5- دمج العلاقات
138	6- لغة تعريف المعطيات DDL
139	6-1- تعريف المجالات بلغة SQL
140	6-2- تعريف المخطط العلاقي بلغة SQL
141	6-3- حذف مخطط علاقة بلغة SQL
142	7- لغة SQL المضمنة
144	تمارين الفصل الرابع
147	الفصل الخامس شروط التكامل
147	مقدمة
148	1- تكامل المجالات
148	2- التكامل المرجعي
149	2-1- مفاهيم أساسية
152	2-2- التكامل المرجعي في لغة SQL
153	3- التأكيد ضمن قاعدة المعطيات
154	4- القادح Trigger
156	5- الارتباطات التابعة Dependencies Functional
156	5-1- مفاهيم أساسية
157	5-2- إغلاق مجموعة من الارتباطات التابعة
159	5-3- خوارزمية لإيجاد المجموعة المغلقة لمجموعة واصفات

160	5-4- التغطية الصغرى للارتباطات التابعة
161	6- الخلاصة
162	تمارين الفصل الخامس
163	الفصل السادس: تصميم قاعدة معطيات علاقاتية
163	مقدمة
166	1- التقييس باستخدام الارتباطات التابعة
166	1-1- خواص التجزئة المستخدمة للتقييس
167	1-2- المنهجية المتبعة في التقييس
167	2- الأشكال النظامية
167	1-2- الشكل النظامي الأول First Normal Form 1NF
168	2-2- الشكل النظامي الثاني Second Normal Form 2NF
170	3-2- الشكل النظامي (BCNF (Boyce-Codd Normal Form
173	4-2- الشكل النظامي الثالث (Third Normal Form)
176	5-2- الشكل النظامي الرابع (Fourth Normal Form)
183	3- التقييس باستخدام الارتباط الدمجي
185	4- الشكل النظامي لمجال المفتاح DKNF
187	الفصل السابع المخطط الداخلي لقاعدة المعطيات
187	مقدمة
187	1- تعاريف

- 188 2- تنظيم التسجيلات
- 189 3- تمثيل الكتل
- 189 4- الفهارس الأساسية
- 190 1-4- الكومة
- 191 2-4- ملفات القطيع
- 191 3-4- الملفات المفهرسة
- 192 5- الفهارس المتعددة المستويات
- 195 الفصل الثامن : المناقلات
- 195 1- تعاريف
- 197 2- الحالات المختلفة للمناقلة
- 201 3- الأقفال (Locks)
- 201 1-3- الأقفال وإدارة الوصول المتزامن إلى عناصر المعلومات
- 202 2-3- بعض مشاكل الأقفال
- 203 4- منسق المناقلات
- 204 5- نموذج مناقلة
- 204 6- خوارزمية تحويل تنفيذ متداخل إلى تنفيذ متسلسل
- 206 7- بروتوكول الإقفال على مرحلتين
- 206 1-7- أقفال القراءة والكتابة
- 207 2-7- اكتشاف العرقلة المتبادلة

209	الفصل التاسع: معالجة الأعطال
209	تعريف
210	1- الاحتياطات الأولية
211	2- الوصول للمعطيات
213	3- معالجة الأعطال
216	المراجع

الفصل الأول

مدخل إلى قواعد المعطيات

مقدمة

تكوّن قواعد المعطيات فرعاً أساسياً من فروع المعلوماتية، وتعتبر المفاهيم التي تعتمد عليها والأدوات التي تقدمها العمود الفقري لتطوير أنواع عديدة من التطبيقات المعلوماتية الواسعة الانتشار، خاصة في التطبيقات الإدارية والمالية والنظم المساعدة في اتخاذ القرار.

بدأ الاهتمام بهذا النوع من التقنيات مع ظهور الحاجة إلى إدارة حجم كبير من المعطيات، إذ تبين عجز طرق البرمجة التقليدية عن مواكبة التطور الحاصل في حجومات هذه المعطيات، وهذا ما استدعى إنشاء نظم عامة تهدف في المقام الأول، إلى تحسين وتسهيل طرق التعامل مع حجومات كبيرة من المعطيات، من قبل عدد كبير من المستخدمين مع تحقيق أمن المعطيات.

تعرف قاعدة المعطيات بأنها "مجموعة من المعطيات المهيكلة غير المتكررة، المسجلة على وسط تخزين يسمح بالوصول إليها من قبل عدة برامج تطبيقية".

لقد كان هذا التعريف عاما في بداية ظهور قواعد المعطيات كفرع مستقل من فروع المعلوماتية وانبثقت عنه مجموعة واسعة من الأهداف التي سعى العاملون في هذا المجال لتحقيقها بتطوير نظم إدارة قواعد المعطيات.

يتألف نظام إدارة قواعد المعطيات (DBMS : Data Base Management Systems) من تجمع من المعطيات المرتبطة فيما بينها، ومجموعة من البرامج التي توفر الوصول إلى هذه المعطيات.

تحوي قاعدة المعطيات عادة معلومات عن مؤسسة ما، وبسبب الأهمية البالغة للمعلومات في المؤسسات فقد برزت الحاجة إلى تطوير مفاهيم ومنهجيات وتقنيات عديدة لإدارة هذه المعلومات إدارة فعالة.

إن الهدف الأساسي لنظم إدارة قواعد المعطيات هو توفير محيط عمل ملائم وفعال يمكن من تخزين المعلومات ضمن قاعدة المعطيات واسترجاعها لاحقاً. وقد صُممت هذه النظم لإدارة كميات ضخمة من المعلومات. وتتضمن إدارة المعلومات المهام الرئيسية التالية :

- تعريف بنى تخزين المعلومات.

- إيجاد التقنيات الملائمة للتعامل مع المعلومات المخزنة.

- تقديم نظم أمان لحماية المعلومات المخزنة من الوصول غير المشروع.

- تجنب التضارب في المعلومات المخزنة نتيجة تشارك عدة مستثمرين في الوصول إلى المعلومات.

نعرض فيما يلي، بأسلوب مبسط، المفاهيم الأساسية المستخدمة في نظم قواعد المعطيات والمبررات التي قادت إلى تطويرها.

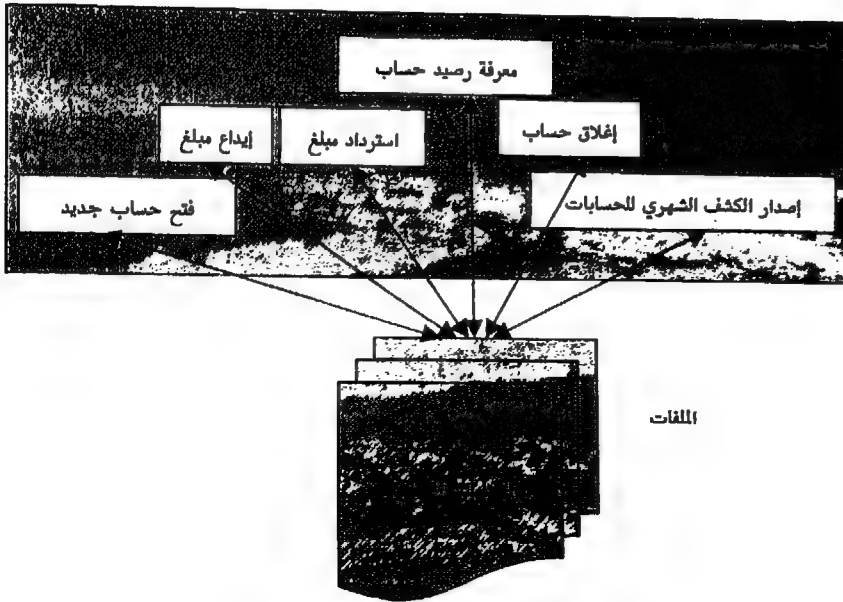
1- الغرض من نظم إدارة قواعد المعطيات

سنبين الغرض من نظم إدارة قواعد المعطيات من خلال مثال لتطبيق معلوماتي بسيط نسبياً، لنبين الصعوبات التي يصادفها مطورو الأنظمة المعلوماتية إذا استخدموا الأدوات التقليدية، مثل نظم إدارة الملفات، في تخزين وإدارة المعطيات، وأهمية نظم قواعد المعطيات في حل هذه الصعوبات.

لنأخذ مؤسسة مصرفية للتوفير، مثل مؤسسة توفير البريد، تحتفظ بمعلومات عن زبائنها وحساباتهم المصرفية في نظام ملفات تقليدي، وتدير هذه المعلومات بواسطة مجموعة من البرامج التطبيقية التي تسفح بإجراء العمليات التالية :

- فتح حساب جديد.
- إيداع مبلغ في حساب مصرفي موجود.
- استرداد مبلغ في حساب مصرفي موجود.
- معرفة رصيد حساب مصرفي.
- إغلاق حساب مصرفي (حذف الحساب)
- إصدار الكشف الشهري للحسابات.

يمكن تمثيل هذا النظام بالشكل (1-1) التالي.



الشكل (1-1) : مثال لنظام معلومات مصرفي

تتعامل البرامج مع مجموعة من الملفات أهمها :

- ملف الحسابات (دليل الحسابات) الذي يتضمن مجموعة من التسجيلات يحوي كل منها معلومات عن حساب واحد مثل رقم الحساب، واسم صاحب الحساب، وعنوانه، وتاريخ فتح الحساب، ... إلخ

• **ملف العمليات المصرفية :** ويتضمن المعلومات المتعلقة بالعمليات المصرفية من إيداع واسترداد. ويمكن أن تتضمن كل تسجيلة من هذا الملف معلومات عن حركة واحدة مثل نوع العملية، وتاريخها، ورقم الحساب، والمبلغ، ... إلخ.

وتتضمن مجموعة الملفات أيضاً ملفات عديدة مثل الملفات المؤقتة التي تستخدم في عمليات الفرز والترتيب، والملفات الدائمة التي تستخدم لأرشفة المعلومات التاريخية مثل الحسابات التي أغلقت. أما البرامج التطبيقية السابقة، فيُفترض أنها كتبت من قبل المبرمجين لسدّ الحاجات الحالية للمؤسسة.

عندما تزداد حاجة المؤسسة، تُكتب برامج تطبيقية جديدة لسدّ هذه الحاجة. لنفترض مثلاً أن قوانين جديدة سمحت للمؤسسة بفتح حسابات مصرفية جارية (شيكات). في هذه الحالة سوف تضاف ملفات جديدة تحوي المعلومات المتعلقة بالحسابات الجارية، وستحتاج المؤسسة إلى مجموعة إضافية من البرامج التطبيقية لإدارة هذه الحسابات. وهكذا نرى أن المؤسسة ستحتاج إلى إضافة العديد من الملفات والبرامج التطبيقية إلى نظامها التقليدي لسدّ حاجاتها المتنامية. سنبين المشاكل الأساسية لهذا الحل التقليدي من خلال الملاحظات التالية :

1-1- تكرار المعطيات وتضاربها

يمكن أن تستغرق عمليات تعريف الملفات وكتابة البرامج التطبيقية مدة طويلة، وقد يعمل في ذلك مبرمجون مختلفون، وقد تظهر مشاكل عديدة : فالملفات ولدت بأشكال مختلفة، ويمكن أن تكون البرامج التطبيقية قد كُتبت بلغات برمجة مختلفة أيضاً. وأكثر من ذلك يمكن أن تكون المعلومات نفسها مكررة في أكثر من ملف. فمثلاً من الممكن أن تكون المعلومات المتعلقة بعنوان ورقم هاتف زبون معين موجودة في ملف الحسابات الجارية، وفي ملف حسابات التوفير بآن واحد.

إن وجود مثل هذا التكرار يسبب هدراً في حجم التخزين، وكلفة عالية في الوصول إلى المعطيات، ويمكن أن يؤدي إلى معطيات متضاربة، وذلك أن وجود عدة نسخ من المعطيات في ملفات مختلفة لا تبقى متوافقة مدة طويلة (حدوث تعديل أو حذف في مكان دون آخر).

فمثلاً، إذا غير أحد الزبائن، الذين يملكون حساب توفير وحساباً جارياً، عنوانه، وجرى تعديل العنوان المسجل في ملف حسابات التوفير دون الانتباه إلى ضرورة تعديل العنوان المخزن في ملف الحسابات الجارية، فإن هذا الزبون لن يستلم كشف حسابه الجاري في بداية الشهر التالي في عنوانه الجديد.

1-2- صعوبة الوصول إلى المعطيات

نفترض أن أحد موظفي المصرف يحتاج إلى معرفة أسماء الزبائن القاطنين في منطقة محددة بغية إجراء دراسة معينة. سوف يقوم بتقديم هذا الطلب إلى قسم المعلوماتية في المصرف. لما كان هذا الطلب جديداً ولم يكن موضوعاً عند تصميم النظام الأساسي، فلا يوجد تطبيق يولد مثل هذه القائمة. ويبقى أمام الموظف حلان لإيجاد مثل هذه القائمة هما : أن يطلب قائمة بأسماء جميع الزبائن المتعاملين مع المصرف مع إقامتهم، ثم يقوم بانتقاء المطلوبين يدوياً، أو أن يطلب من أحد المبرمجين كتابة البرنامج التطبيقي اللازم لذلك. إن كلا الحلين غير كاف، فمن الممكن أن تتولد حاجة أخرى لدى الموظف بعد عدة أيام تعيده إلى نفس المشكلة.

يتبين من هذا المثال أن محيط إدارة الملفات لا يسمح باسترجاع المعطيات المطلوبة بطريقة فعالة، ولا بد من كتابة برامج عديدة لمعالجة الاستفسارات المختلفة. لذلك يصبح من الضروري تطوير نظام عام يفيد في استرجاع المعطيات استرجاعاً أفضل.

1-3- عزل المعطيات

لما كانت المعطيات موزعة في عدة ملفات ذات بنى مختلفة، فإنه من الصعب كتابة تطبيق جديد لاسترجاع المعطيات وفق أشكال معينة. إن سبب هذه المشكلة هو أن تعريف المعطيات يجري ضمن البرامج التي تدير هذه المعطيات. ومن ثم فإن أي تعديل في بنى التخزين يجب أن يواكبه تعديل كل البرامج التي تتعامل مع المعطيات.

1-4- تعارض في الوصول المتزامن

تسمح النظم المعلوماتية الكبيرة لأكثر من مستخدم بالوصول إلى المعطيات لإجراء عمليات الإضافة والحذف والتعديل والاستفسار. وذلك بغية الحصول على زمن استجابة أقصر وزيادة مردود هذه النظم. ولكن ذلك يزيد من أخطار التعديل المتزامن للمعطيات ويزيد تضارب المعطيات الناتجة عنه. سنوضح ذلك بالمثال التالي :

ليكن لدينا حساب مصرفي A يحوي 5000 ل.س وهناك مستخدمان يقوم الأول بتسجيل عملية استرداد 500 ل.س والثاني يسجل عملية إيداع 1000 ل.س في الوقت نفسه من الحساب A. فإذا حدثت العمليتان متزامنتين، أمكن أن تتركبا نتيجة الحساب في حالة خاطئة، قد تكون 4500 ل.س أو 6000 ل.س بدلاً من 5500 ل.س*.

* تنشأ هذه المشكلة بسبب إجراء نسخ من التسجيلات المراد تعديلها إلى الذاكرة المركزية، وإجراء التعديلات المطلوبة على النسخة الموجودة في الذاكرة المركزية، ثم إعادة كتابة القيم الجديدة في الملف. فإذا قام أحد التطبيقين بإجراء عملية النسخ قبل أن ينهي التطبيق الآخر التعديل والكتابة فإنه يكون قد قرأ معطيات في طور التعديل.

لحل هذه المشكلة جرى تطوير طرق مختلفة لمراقبة هذا النوع من العمليات. ولكن لما كان من الممكن أن تكون المعطيات مستخدمة في عدة تطبيقات، فإن إيجاد أشكال موحدة وموثوقة للمراقبة يصبح أمراً صعباً جداً.

1-5- أمن المعطيات

يُقصد بأمن المعطيات قدرة النظام على تحديد صلاحيات الوصول إلى المعطيات. فمثلاً في النظام المصرفي يحتاج المسؤول عن دفع رواتب موظفي المصرف إلى معرفة معلومات عن موظفي المصرف، ولا يحتاج إلى معرفة معلومات عن الحسابات والزبائن المتعاملين مع المصرف. لتحقيق مثل هذه الإمكانيات تحتاج طرق البرمجة التقليدية إلى إضافة تطبيقات عديدة إلى النظام. وفي بعض الأحيان يحتاج تحقيق بعض شروط الأمن إلى جهد يتجاوز الجهد المبذول في تحقيق الوظائف الأساسية للنظام.

1-6- تكامل المعطيات

قد تخضع المعطيات المخزنة في النظام لشروط مغينة. فمثلاً يمكن أن يشترط المصرف أن رصيد أي حساب يجب ألا يقل عن 250 ل.س، وأن الرصيد الأعظم لحساب التوفير هو مليون ليرة سورية. وينبغي أخذ مثل هذه الشروط بعين الاعتبار في جميع البرامج التطبيقية التي يتضمنها النظام. وكما نرى فإنه من الصعب جداً في نظام إدارة الملفات إضافة شرط جديد إلى المعطيات، لأن ذلك يتطلب تعديل جميع البرامج المكتوبة سابقاً، والتي تستخدم هذه الملفات، كما أنه من الصعب إضافة شروط متعلقة بمعطيات مختلفة مخزنة في ملفات مختلفة.

2- الوظائف التي توفرها أنظمة إدارة قواعد المعطيات

لحل المشاكل المذكورة آنفاً، ولتوفير إمكانات إضافية، جرى تطوير نظم إدارة قواعد المعطيات كطريقة عامة، تشمل مجموعة من المفاهيم وتوفر برمجيات عامة تفيد في تحقيق الأهداف التالية :

2-1- مركزية المعلومات

تهدف قواعد المعطيات إلى تجميع كافة المعطيات المتعلقة بمؤسسة ما ضمن نظام واحد، يقوم بإدارة هذه المعطيات إدارةً قياسية، ويوفر جميع حاجات التطبيقات من المعطيات. يوفر اعتماد نظام معلومات مركزي في أي مؤسسة مزايا عديدة أهمها إلغاء التكرار، وتوفير سهولة إدخال وتحديث المعلومات، ومركزية التحكم والمراقبة.

2-2- استقلال المعطيات

الهدف الأساسي لنظم إدارة قواعد المعطيات هو توفير الوسائل الكفيلة بجعل المعطيات مستقلة عن طريقة التخزين وعن البرامج التي تقوم بالتعامل مع هذه المعطيات. يجري تحقيق هذه الغاية بجعل التعامل مع المعطيات بواسطة برامج تقوم بالوصول إلى هذه المعطيات من مستوى عال من التجريد، لا يظهر الطريقة الفعلية للتخزين، ولا يحتاج إلى معرفة كافة التفاصيل المتعلقة ببنية القاعدة ومحتوياتها الشاملة.

إن تحقيق هذا الهدف بواسطة البرامج التي يتضمنها نظام إدارة قواعد المعطيات يخفف الأعباء الملقاة على عاتق المبرمجين، والمتمثلة في ضرورة تعديل البرامج التطبيقية لدى كل تعديل في بنى تخزين المعطيات، سواء في بنية الملفات أو في طريقة تنظيم الملفات أو وسائط التخزين. تتيح نظم إدارة قواعد المعطيات إمكاناً للتعامل مع المعطيات بقطع النظر

عن بنيتها الداخلية، وتمكن البرامج التطبيقية من متابعة العمل على المعطيات، في حال حدوث تغير في بني التسجيلات لا يتناقض مع البنى التي كانت تستخدمها، ولا تتأثر البرامج التطبيقية بتغير طرق الوصول. فإذا جرت إضافة فهرس Index ، أو دمج ملفان في ملف واحد، فإن ذلك لا يستدعي تغيير البرامج التي تدير المعطيات.

2-3- معالجة المعطيات بواسطة لغات غير إجرائية

معظم مستثمري نظم إدارة قواعد المعطيات هم مستثمرون عاديون ليس لديهم فكرة سابقة عن لغات البرمجة. لذلك يجب توفير لغة يستطيع المستثمر بواسطتها أن يسأل قاعدة المعطيات أو يعدّل تلك المعطيات دون تحديد خوارزمية الوصول إلى تلك المعطيات، بل فقط بأن يصف المعطيات التي يريد المستثمر التعامل معها. يسمى هذا النوع من اللغات لغات غير إجرائية.

تعتبر هذه النقطة من أهم الأهداف التي يجب أن يحققها نظام إدارة قواعد المعطيات، فوجود لغة غير إجرائية عالية المستوى يسمح لأي مستثمر كان بأن يستفيد من إمكانيات النظام بفعالية و سهولة.

2-4- التسهيلات الخاصة بإدارة المعطيات

توفر نظم إدارة قواعد المعطيات الوسائل اللازمة للتعبير عن المعطيات (طريقة تعريفها وتخزينها) والوصول إليها وعرضها. تسمى هذه الوسائل أدوات إدارة قواعد المعطيات. وللحصول على إدارة فعالة وجيدة للمعطيات، يجري عادة حصر بعض هذه الأدوات بشخص واحد يدعى مدير النظام أو بعدة أشخاص يملكون امتيازات خاصة.

2-5- الوصول إلى المعطيات بفعالية

تسعى أنظمة إدارة قواعد المعطيات لزيادة عدد الإجراءات التي تنفذ في ثانية واحدة. وذلك بزيادة عدد المستخدمين الذين يستطيعون الوصول إلى المعطيات بآن واحد، وإنقاص زمن الاستجابة (الزمن اللازم للحصول على جواب طلب ما). لتحقيق ذلك تتضمن هذه الأنظمة خوارزميات وطرقاً خاصة لتقسيم المصادر (الوحدة المركزية، وحدات الدخل/الخروج) بين المستثمرين تقسيماً عادلاً.

2-6- التحكم في تكرار المعطيات

إن وجود إدارة مركزية للمعطيات تمكن من حل مشكلة تكرار المعطيات، وذلك بإعطاء الانطباع بأن كل مستخدم من مستخدمي قاعدة المعطيات يتعامل مع نسخة مستقلة من قاعدة المعطيات، وتوفير الأدوات التي تنسق بين العمليات التي يجريها المستخدمون على النسخة الوحيدة من المعطيات.

2-7- تكامل المعطيات

يسمح نظام إدارة قواعد المعطيات بتحقيق أنواع عديدة من شروط التكامل نعرضها من خلال أمثلة :

- تكامل وحدات المعطيات. مثال : لا يمكن فتح حساب مصرفي لزيد دون معرفة عنوانه
- التكامل المرجعي. مثال : لا يمكن إجراء عمليات مصرفية على حساب قبل فتح الحساب
- وشروط التكامل المعرفة من قبل المستخدم : مثال : الرصيد أكبر من 250 ل.س.

لتحقيق ذلك يوفر نظام إدارة قواعد المعطيات الإمكانيات اللازمة لتعريف هذه الشروط من جهة، ولكشف وإيقاف جميع العمليات التي قد تؤدي إلى الإخلال بهذه الشروط من جهة أخرى.

2-8- تقسيم المعطيات

يُقصد بتقسيم المعطيات السماح بتقسيم معطيات قاعدة ما بين عدة تطبيقات، بحيث يستطيع كل منها الوصول إلى المعطيات دون أن ينتظر تطبيقاً آخر.

2-9- أمن المعطيات

توفر نظم إدارة قواعد المعطيات إمكان حماية بعض المعطيات الخاصة، بحيث أن مجموعة محددة هي فقط التي تستطيع الوصول إلى تلك المعطيات. فمثلاً لا يستطيع مدير قسم معين أن يطلع على رواتب كل العاملين في الشركة، بل على رواتب الموظفين العاملين في قسمه فقط.

3- تصميم قواعد المعطيات

من وجهة النظر البرمجية، تتألف قاعدة المعطيات من تجمع من الملفات المترابطة، ومجموعة من البرامج التي تسمح بالوصول إلى المعطيات المخزنة فيها واسترجاعها وتعديلها. وتهدف نظم قواعد المعطيات إلى تقديم إمكان التعامل مع جزء من المعطيات واستخدامها بطريقة فعالة تجعلها بمتناول عدد كبير من المستخدمين.

تقود هذه الاعتبارات إلى تصميم بنى معطيات معقدة لتمثيل المعطيات، مع ضرورة إخفاء هذا التعقيد للسماح لأكبر عدد من المستخدمين، الذين لا يملكون خبرة واسعة في البرمجة، بالوصول إلى المعطيات.

لتحقيق ذلك، ولما كان مستخدمو قاعدة المعطيات ليسوا بالضرورة خبراء في استخدام الحواسيب والبرمجة، فإنه يجب إخفاء التعقيد الموجود في تلك البنى بتحقيق وجود عدة مستويات للتصميم تتوافق مع مستوى التفصيل الذي يمكن لكل فئة من المستخدمين التعامل معه.

لتحقيق هذه الأهداف جرى تحديد ثلاثة مستويات من التجريد تسمى مخططات (Schema) لتوصيف أي قاعدة معطيات. يجري في كل مستوى توصيف القاعدة ببعض التفصيل الإضافي، عن المستوى الأعلى، كما يقدم نظام إدارة قواعد المعطيات الوسائل الكفيلة بإيجاد الترابط بين هذه المستويات المختلفة. تهدف هذه النماذج إلى تبسيط تعامل المستخدمين مع المعطيات. ونبين فيما يلي شرحاً مبسطاً لهذه المستويات :

- المخطط المفاهيمي : يعتبر المخطط المفاهيمي تجريداً للواقع يعكس عناصر المعلومات التي ستقوم قاعدة المعطيات بإدارتها. يجري من خلال المخطط المفاهيمي توصيف المحتوى المعلوماتي للقاعدة دون التعرض لأساليب النمذجة اللاحقة أو للاستفسارات التي سيجريها المستخدمون. يسمح هذا الفصل بترجمة المخطط المفاهيمي إلى أنواع مختلفة من المخططات المنطقية.

يجري التعبير عن المخطط المفاهيمي بأشكال عديدة، وتعتبر مخططات الكيانات والارتباطات (ERD : Entity-Relationship Diagrams) أحد أهم الطرق المتبعة في إنشاء مخطط المفاهيمي. يسمح هذا النموذج بتوصيف قاعدة المعطيات بشكل مخططات بيانية تتضمن الكيانات الداخلة في بنية النظام والارتباطات بينها.

• المستوى المنطقي : وصف لمحتوى قاعدة المعطيات بلغة قياسية تمكن مطوري التطبيقات ومستخدمي القاعدة من التعامل مع المعطيات على مستوى عالٍ من التجريد، يجنبهم الخوض في التفاصيل المتعلقة ببنية الملفات وطرق الوصول إليها. يعتبر المخطط المنطقي نواة قاعدة المعطيات، وتمثيلاً معيارياً لمعطيات المؤسسة يعكس طبيعة المعطيات وخصائصها وارتباطاتها. في هذا المستوى من التجريد يجري تحديد مايلي:

- أنماط المعطيات البسيطة والمركبة المستخدمة في المؤسسة.
- ارتباطات هذه الأنماط بعضها ببعض، بما يعكس واقع عمل المؤسسة.
- قواعد تكامل المعطيات: الخصائص التي يجب أن تحققها المعطيات المخزنة في القاعدة.
- يوفر كل نظام إدارة قواعد معطيات على الأقل. نموذجاً للمعطيات، يسمح للمستخدمين بالتعامل مع المعطيات بأسلوب أقرب ما يكون إلى الواقع الذي أتت منه هذه المعطيات. يتضمن أي نموذج مفهومي أساسيين:
- طريقة تعريف المعطيات.
- العمليات التي يمكن تطبيقها على المعطيات.

في النموذج العلاقتي ، مثلاً ، يمكن تعريف المعطيات بأنها مجموعة من العلاقات ، وكل علاقة هي جدول يحوي عدداً من الأسطر والأعمدة . يحوي كل عمود قيمة تنتمي إلى مجال معين .

توفر هذه النماذج من المعطيات إمكان النظر إلى المعطيات من مستوى أعلى دون الخوض في طريقة التخزين الفيزيائي أو طرق الفهرسة ، وتترك هذه المهام لنظام إدارة قواعد المعطيات الذي يقوم بإيجاد مايقابل هذه البنى في المستوى الفيزيائي . في نظم إدارة قواعد المعطيات ،

يجري التعبير عن بنية قاعدة المعطيات بواسطة لغة عالية المستوى تسمى لغة تعريف المعطيات (DDL). تسمح هذه اللغة إضافةً إلى ماسبق بتعريف شروط تكامل المعطيات.

• المستوى الفيزيائي (الداخلي) **Physical level** : وهو المستوى الأدنى في تجريد المعطيات، ويصف الطريقة الفعلية لتخزين المعطيات. يتعلق المستوى الداخلي ببنية التخزين التي ستحتوي المعطيات فعلياً، ويسمح بوصف المعطيات حسب الطريقة المتبعة في التخزين:

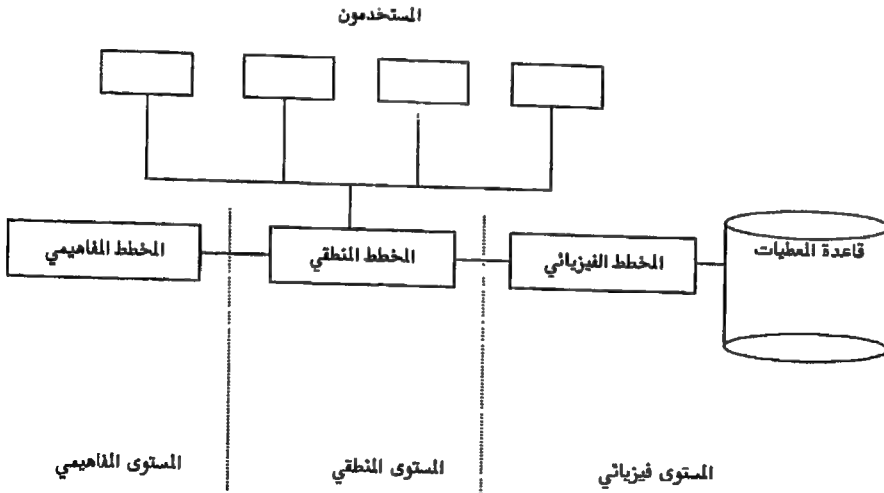
• توصيف ملفات قاعدة المعطيات (أنماط الحقول و أطوالها، الحقول المركبة، ... الخ).

• طرق التعامل مع وسط التخزين (**Segments, Blocks, Buffers**) ، ... الخ) .

• طرق الوصول إلى التسجيلات (الفهرسة ، ربط التسجيلات ، ... الخ).

غالباً لا يحتاج القارئون على إدارة قواعد المعطيات إلى التدخل على هذا المستوى، ويتركون هذه المهمة لنظام إدارة قواعد المعطيات الذي يقوم بترجمة النموذج المنطقي إلى نموذج فيزيائي مكافئ.

في معظم الأحيان يجري تعريف مجموعات جزئية من المعطيات، تتضمن كل منها الجزء الذي يهم مستمراً معيناً أو فئة من المستثمرين. تسمى كل مجموعة من هذه المجموعات مخططاً خارجياً.



الشكل (1-2) : مستويات توصيف قاعدة المعطيات

4- نماذج المعطيات Data Models

يُقصد بنمذجة المعطيات، في قواعد المعطيات، استخدام مجموعة الأدوات التصميمية التي تساعد على وصف المعطيات، والعلاقات المتبادلة فيما بينها، ودلالة المعطيات، وشروط تناسقها.

تقسم نماذج المعطيات إلى ثلاث مجموعات :

- النماذج المنطقية المعتمدة على الأغراض **Object-based logical models**
- النماذج المنطقية المعتمدة على التسجيلات **Record-based logical models**
- نماذج المعطيات الفيزيائية **Physical models**

4-1- النماذج المنطقية المعتمدة على الأغراض

تُستخدم في وصف المعطيات في المستويين المفاهيمي والمنطقي (الخارجي). وتمتاز بأنها تسمح ببنية مرنة وبتحديد شروط على المعطيات بوضوح. توجد نماذج مختلفة متعددة ومعروفة منها :

- نموذج الكيانات والارتباطات Entity-Relationship Model

- النموذج الغرضي التوجه Object-Oriented Model

- النموذج الثنائي Binary Model

- النموذج الدلالي Semantic Data Model

- النموذج المنطقي الدلالي Info-logical Model

- النموذج الوظيفي Functional Data Model

سوف ندرس فيما يلي نموذج الكيانات والارتباطات والنموذج الغرضي التوجه باعتبارهما مثال للنماذج المنطقية المعتمدة على الأغراض.

4-1-1- نموذج الكيانات والارتباطات

يعتمد هذا النموذج على إدراك العالم الحقيقي المؤلف من مجموعة من الأغراض الأساسية المسماة كيانات (Entities) والعلاقات فيما بينها أي الارتباطات (Relationships).

الكيان : هو غرض يتميز من الأغراض الأخرى بتحديد مجموعة من الواصفات المميزة. فمثلاً (رقم الحساب ، الرصيد) يميزان حساباً محدداً في المصرف.

الارتباط : هي علاقة ربط بين عدة كيانات. مثال : علاقة زبون-حساب تربط كل زبون بحسابه المصرفي.

تسمى مجموعة الكيانات التي لها نفس النوع صف الكيانات، وتكون مجموعة الارتباطات من نفس النوع ما يسمى بصف الارتباط.

وإضافة إلى تمثيل الكيان والارتباطات، يتيح نموذج E-R تمثيل شروط إضافية يجب أن يحققها محتوى قواعد المعطيات. أحد أهم هذه الشروط هو (درجة الارتباطات Mapping Cardinalities) التي تعبر عن عدد الكيانات التي يمكن أن ترتبط بكيانات أخرى بواسطة الارتباط المعروف فيما بينها.

4-1-2- النموذج الغرضي التوجه

يرتكز هذا النموذج، مثل نموذج الكيانات والارتباطات، على الأغراض. يحوي كل غرض قيماً مخزنة في متحولات حالة Instance-Variables داخل الغرض. هذه القيم هي نفسها أغراض، وبذلك يحوي الغرض أغراضاً وبشكل شجري (مستوى عمق شجري من الشبكة). كما يحوي الغرض مجموعة من البرمجيات تعمل لهذا الغرض تُسمى طرقاً (Methods). تُجمع الأغراض التي لها نفس نوع القيم ونفس الطرق بعضها إلى بعض فيما يُسمى بصفوف. ويمكن أن يستخدم الصف كنوع لتعريف الأغراض.

إن الجمع بين المعطيات والبرمجيات في تعريف نوع، مشابه لمفهوم أنماط المعطيات المجردة المستخدم في لغات البرمجة. في هذا النموذج، ثمة مستويان من تجريد المعطيات :

المستوى الأول يتضمن طريقة وصول غرض إلى معطيات غرض آخر حيث تجرى مناقشة طريقة الغرض الآخر وهذا ما يُسمى "إرسال رسالة" إلى الغرض، ومن ثم استدعاء واجهة الطرق المتعلقة بغرض، والتي تُعرف الجزء الخارجي المرئي من الغرض.

المستوى الثاني من التجريد يتعلق بالجزء الداخلي من الغرض، وهو متحولات الحالة وبرمجيات الطرق، وهي غير مرئية من الخارج.

لتوضيح هذا المفهوم نأخذ مثال النظام المصرفي، ولنعتبر غرضاً مثل الحساب المصرفي. متحولات الحالة في هذا الغرض هي : رقم الحساب، والرصيد. الطرق المربوطة بهذا الغرض هي دفع الفوائد pay-interest التي تقوم بإضافة الفائدة إلى الرصيد. ولنفترض أن الفائدة على جميع الحسابات في المصرف هي 6٪ ويود المصرف أن يغير الفائدة لتصبح 5٪ للحساب ذي الرصيد الذي ينقص عن 10000 ل.س و6٪ للحسابات الأخرى. يتطلب تفعيل هذا التغيير في معظم نماذج المعطيات تفعيل تغيرات في برمجيات تطبيق أو أكثر في النظام، ولكن تفعيله في النموذج الغرضي التوجه يتطلب فقط تغييراً في طريقة دفع الفوائد وتبقى الواجهة الخارجية للغرض دون أي تغيير.

4-2- النماذج المنطقية المعتمدة على التسجيلات

تُستخدم هذه النماذج، كالنماذج المعتمدة على الأغراض، في وصف المعطيات في المستويين المفاهيمي والخارجي وتحديد البنية المنطقية لقاعدة المعطيات وتقديم وصف بمستوى عالٍ للتنفيذ.

سميت النماذج بهذا الاسم لأن قاعدة المعطيات منظمة في أشكال ثابتة من التسجيلات. يُعرف كل نوع من التسجيلات بعدد محدد من الحقول والواصفات، وعادةً يكون لكل

حقل طول ثابت. إن استخدام أطوال ثابتة في التسجيلات يساعد على تبسيط مستوى تمثيل التغيرات في قاعدة المعطيات.

لا تحوي النماذج المنطقية المعتمدة على التسجيلات تقنيات لتمثيل البرمجة المباشرة في قاعدة المعطيات. وعوضاً عن ذلك، توجد لغة منفصلة مرتبطة بالنموذج للتعبير عن الاستفسارات وإجراء التعديل في قاعدة المعطيات.

من النماذج المنطقية المنتشرة المعتمدة على التسجيلات النموذج الشبكي، والنموذج الهرمي والنموذج العلاقتي. وقد انتشر النموذج العلاقتي انتشاراً واسعاً في السنوات الأخيرة، على حين استخدم النموذجان الهرمي والشبكي في قواعد المعطيات القديمة نسبياً.

سنورد فيما يلي شرحاً مختصراً لهذه النماذج، وسندرس النموذج العلاقتي دراسة مفصلة في فصل مستقل.

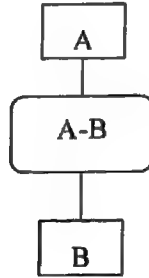
4-2-1- النموذج الشبكي (Network Model)

تمثل المعطيات في هذا النموذج كتجمع لتسجيلات لها بنية كما في لغة Pascal أو لغة PL/I، وتمثل العلاقات بين المعطيات بروابط يُعبر عنها بمؤشرات. تُنظم التسجيلات في قاعدة المعطيات تنظيمًا اعتباطيًا.

قاعدة المعطيات : بيان عقده تسجيلات تتصل بمؤشرات منطقية.

نمط التسجيلات (العناصر) : تحوي التسجيلات معطيات ثابتة (مثل av#, cap) وعدة أنواع من البنى (الشعاع، وهو مجموعة من العناصر التي لها نفس النوع، المجموعة Coset).

البنية الأساسية في النموذج الشبكي هو المجموعة المتممة Coset التي تعبر عن ارتباط تبعية، حيث يوجد في كل تسجيلية من نوع A (الأب أو المالك) مجموعة تسجيلات من نوع B (الأبناء أو الأعضاء).



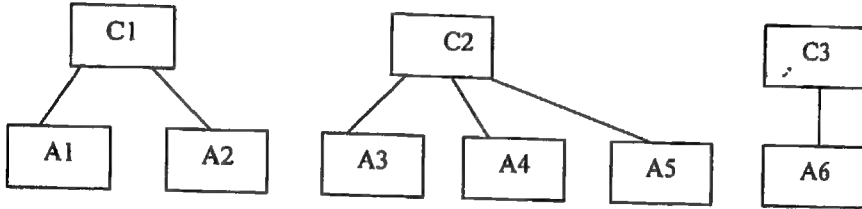
الشكل (3-1) : البنية الأساسية للنموذج الشبكي

مثال لمحتوى قاعدة معطيات نظام مصرفي (customer, account)

```

Type Customer = record
    Customer_name : string;
    Customer_street : string;
    Customer_city : string;
End;
Account = record
    Account_number : string;
    Account_balance : string;
End;
  
```

يمكن تمثيل محتوى القاعدة كمايلي :

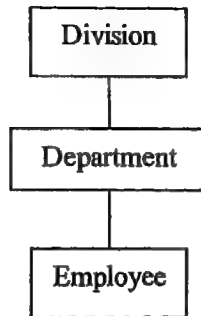


الشكل (3-1) : محتوى قاعدة في النموذج الشبكي

4-2-2- النموذج الهرمي

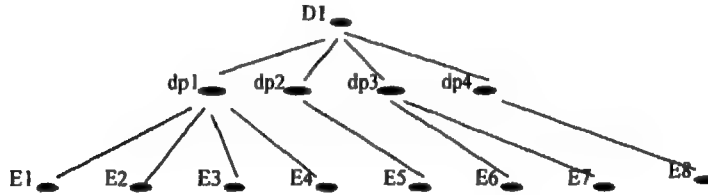
هو مشابه للنموذج الشبكي من حيث تمثيل المعطيات والعلاقات فيما بينها، ولكنه يختلف عنه بأن التسجيلات منظمة في قاعدة المعطيات، ومؤلفة من مجموعة أشجار تمثل ارتباطات هرمية.

مثال أول : التمثيل الهرمي لمؤسسة لها عدة فروع يحوي كل فرع عدداً من الأقسام، يعمل في كل قسم مجموعة من العاملين.



الشكل (4-1) : تمثيل البنية العامة لقاعدة معطيات بالنموذج الهرمي

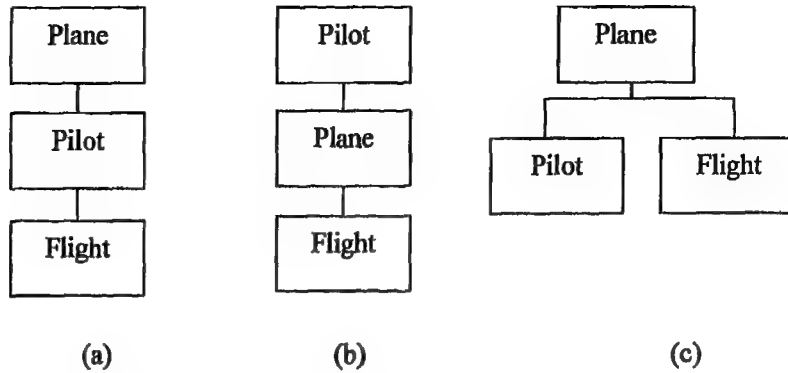
محتوى القاعدة :



الشكل (5-1) : محتوى قاعدة معطيات بالنموذج الهرمي

مثال 2 : شركة طيران يعمل فيها عدد من الطيارين، وتملك عدداً من الطائرات وتقوم بعدة رحلات.

هناك عدة أشكال ممكنة لمخطط القاعدة :



الشكل (6-1) : محتوى قاعدة معطيات بالنموذج الهرمي

4-2-3- النموذج العلاقتي

يسمح بتمثيل المعطيات والعلاقات فيما بينها باستخدام مجموعة من الجداول. يتألف كل جدول من عدد من الأعمدة لكل منها اسم وحيد.

4-2-4- الفرق بين النماذج المختلفة السابقة

يختلف النموذج العلاقتي عن النموذجين السابقين بأنه لا يستخدم المؤشرات أو الروابط، وبدلاً من ذلك يقوم بربط التسجيلات بعضها ببعض عن طريق القيم المحتواة في هذه التسجيلات. يسمح التحرر من استخدام المؤشرات بتعريف أسس لأشكال رياضية مرتبطة بالنموذج العلاقتي.

4-3- النماذج الفيزيائية

تستخدم هذه النماذج لوصف المعطيات في المستوى الأدنى، ويعكس نماذج المعطيات المنطقية فإن عدد نماذج المعطيات الفيزيائية المستخدمة قليل وأشهر اثنين منها هما :

- Unifying Model

- Frame Memory

5- المخططات والحالات

تتبدل المعطيات المخزنة في قاعدة المعطيات مع الزمن بسبب حركة المعلومات المضافة والمحذوفة منها وإليها. ونسمي حالة قاعدة المعطيات **Instance of the Data Base** مجموعة المعلومات المخزنة في قاعدة المعطيات في لحظة معينة. كما نسمي مخطط قاعدة المعطيات **Data Base Schema** القالب العام لقاعدة المعطيات.

ونلاحظ أن مفهوم مخطط قاعدة المعطيات يشبه مفهوم تعريف الأنماط في لغات البرمجة التقليدية، وأن مفهوم حالة لمخطط قاعدة المعطيات يشابه مفهوم قيمة متحول في هذه اللغات.

6- استقلال المعطيات

يُتَصد باستقلال المعطيات إمكان تعديل تعريف مخطط من مخططات قاعدة المعطيات، دون أن يؤثر هذا التعديل في تعريف المخطط من المستوى الأعلى في القاعدة.

ثمة نوعان من استقلال المعطيات :

6-1- استقلال المعطيات فيزيائياً

ويعني أن تعديل المخطط الفيزيائي (طريقة التخزين الفعلية وطرق الوصول إلى المعطيات) لا يتطلب إعادة كتابة البرامج التي تتعامل مع قاعدة المعطيات. والجدير بالذكر أننا نادراً ما نجري تعديلات في المستوى الفيزيائي، إلا في حالات معينة وبقصد تحسين الأداء.

6-2- استقلال المعطيات منطقياً

ويعني إمكان تعديل المخطط المفاهيمي لقاعدة المعطيات دون أن يتطلب ذلك إعادة كتابة البرامج التطبيقية. ويجري تعديل المخطط المفاهيمي عادةً عند تعديل البنية المنطقية لقاعدة المعطيات.

إن تحقيق الاستقلال المنطقي أصعب بكثير من تحقيق الاستقلال، وذلك لاعتماد البرامج التطبيقية إلى حد بعيد على البنية المنطقية للمعطيات المراد الوصول إليها.

7 - لغات قواعد المعطيات

يوفر كل نظام إدارة قواعد معطيات على الأقل لغة واحدة تتيح لمستخدميه تعريف بنية قاعدة المعطيات، وشروط تكامل المعطيات وصلاحيات الوصول إلى المعطيات، وغيرها من التعاريف التي لا بد منها لدى إنشاء القاعدة. وتوفر هذه اللغات طيفاً واسعاً من التعليمات التي تتيح للمستخدم والمبرمج إجراء عمليات الإضافة والحذف والتعديل والاستفسار. وتوفر أنظمة إدارة قواعد المعطيات أدوات خاصة بالتطوير تمكن من تعريف واجهات التعامل مع قاعدة المعطيات (استمارات إدخال، لوحات تحكم، واجهات للاستفسار، تقارير، ... إلخ).

7-1- لغة تعريف المعطيات: (DDL : Data Definition

(Language

يحدد مخطط قواعد المعطيات بمجموعة من التعاريف التي يُعبر عنها بلغة خاصة تسمى لغة تعريف المعطيات. إن نتيجة ترجمة تعليمات هذه اللغة هي مجموعة من الجداول المخزنة في ملفات خاصة تسمى قاموس المعطيات (Data dictionary).

قاموس المعطيات هو ملف يحوي معطيات سامية (meta-data) أي "معطيات عن المعطيات" ويجري استدعاء هذا الملف قبل قراءة أو تعديل المعطيات الحقيقية في نظم قواعد المعطيات.

تحدد بنى التخزين وطرق الوصول المستخدمة من قبل نظم قواعد المعطيات بمجموعة تعاريف في نوع خاص من لغة تعريف المعطيات، تسمى لغة تعريف وتخزين المعطيات.

إن ترجمة هذه التعاريف تعطي مجموعة تعليمات تحدد التفاصيل التنفيذية لمخططات قواعد المعطيات التي لا تظهر للمستثمرين العاديين.

7-2- لغة التعامل مع المعطيات

(DML : Data Manipulation Language)

هي لغة تسمح للمستثمرين بالوصول والتعامل مع المعطيات المنظمة بنموذج معطيات معين. توفر لغة التعامل مع المعطيات الوظائف التالية:

- استخلاص المعطيات المخزنة في قواعد المعطيات
- إضافة معلومات جديدة إلى قاعدة المعطيات
- حذف معلومات من قاعدة المعطيات

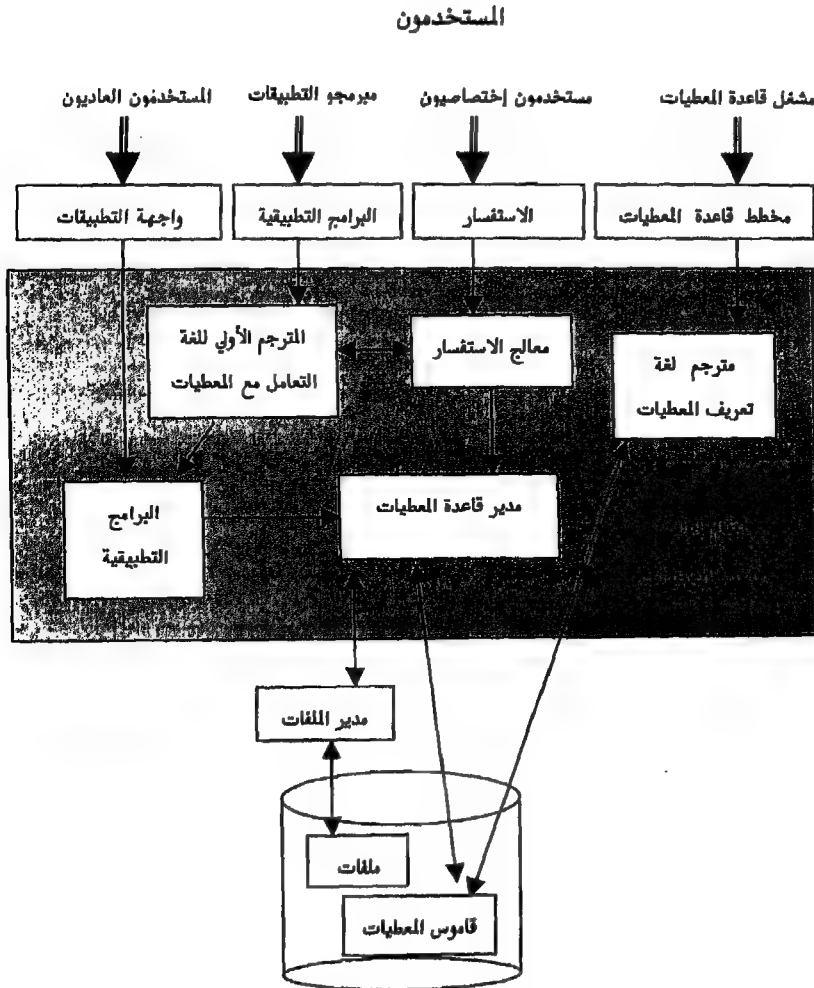
يوجد نوعان رئيسيان من لغات التعامل مع المعطيات:

- لغات إجرائية: وتتطلب من المستثمر تحديد المعطيات التي يحتاج إليها وطريقة الحصول عليها.
- لغات غير إجرائية: وتتطلب من المستثمر تحديد المعطيات التي يحتاج إليها دون تحديد كيفية الحصول عليها.

اللغات غير الإجرائية أسهل تعلماً واستخداماً من اللغات الإجرائية. ولكن لأن المستثمر لا يحدد كيفية الحصول على المعطيات، فيمكن أن تولد اللغة طريقة للوصول ليست فعالة مثل التي يقوم بوضعها المستثمر بلغة التعامل الإجرائية. هذه الصعوبات يمكن أن نتجاوزها باستخدامنا لتقنيات اختزال متعددة.

8- البنية العامة لقاعدة المعطيات

يبين الشكل (7-1) البنية العامة لقاعدة معطيات. وفيما يلي شرح مبسط لأهم العناصر الداخلة في عمل قاعدة المعطيات.



الشكل (7-1) : البنية العامة لقاعدة معطيات

8-1- الاستفسار

هو طلب استخلاص معلومات. ويسمى الجزء المتعلق باستخلاص المعلومات من لغة التعامل مع المعطيات لغة الاستفسار. وفي بعض المصادر يلاحظ استخدام التعبيرين "لغة الاستفسار" و "لغة التعامل مع المعطيات" كمترادفين، وهذا خطأ طبعا.

8-2- مدير قاعدة المعطيات Data base Manager

تتطلب قواعد المعطيات تخزين قدر كبير من المعطيات، ويصل حجم التخزين عادة إلى عدة جيجابايت أو إلى تيرابايت ($1\text{GB}=1000\text{MB}$) و ($1\text{TB}=1000\text{GB}$). وبسبب تعذر تخزين هذه المعلومات في الذاكرة الرئيسية للحاسوب، يجري تخزينها على أقراص. وتنتقل المعطيات بين قرص التخزين والذاكرة الرئيسية عند الحاجة. ولما كانت حركة المعطيات من وإلى القرص بطيئة نسبةً إلى سرعة وحدة المعالجة المركزية، فإن نظم قواعد المعطيات تنظم المعطيات بطريقة تقلل الحاجة إلى حركة المعطيات بين القرص والذاكرة الرئيسية.

الهدف من نظم قواعد المعطيات هو تسهيل وتبسيط الوصول إلى المعطيات. ومستثمرو النظام لايهتمون بالضرورة بالتفاصيل التقنية لتنفيذ النظام، وإنما بأداء النظام. فكلما كان زمن الاستجابة طويلاً كلما تضاءلت قيمة النظام.

يعتمد أداء النظام على فعالية بنية المعطيات المستخدمة لتمثيل المعطيات في قاعدة المعطيات وقدرة النظام على العمل على تلك البنى بفعالية.

مدير قواعد المعطيات هو برنامج يلعب دور الواجهة بين المعطيات المخزنة في المستوى الأدنى من قواعد المعطيات والبرامج التطبيقية والاستفسارات التي يتلقاها النظام. وهو المسؤول عن تنفيذ المهام التالية:

8-2-1- التفاعل مع مدير الملفات

تخزن المعطيات على القرص باستخدام نظم الملفات الموجودة في نظام التشغيل. ويقوم مدير قواعد المعطيات بنقل التعليمات المختلفة للتعامل مع المعطيات إلى المستوى الأدنى من تعليمات نظام الملفات. وبذلك يكون مدير قواعد المعطيات مسؤولاً عن التخزين الفعلي لمعطيات القاعدة، والوصول إليها، وتعديلها.

8-2-2- التحقق من تكامل المعطيات

ينبغي أن تحقق القيم المخزنة في قاعدة المعطيات أنواعاً معينة من شروط التكامل. ويعبر المسؤول عن النظام عن بعض هذه الشروط بشكل واضح وصريح. وبالتالي يجب على مدير قاعدة المعطيات أن يستطيع تحديد إذا كانت قاعدة المعطيات الناتجة من جراء أي تعديل موافقة للشروط. وفي حال عدم تحقق ذلك فيجب القيام بمجموعة من الإجراءات الخاصة (مثل رفض التعديل أو الإدخال).

8-2-3- التحقق من الأمان

قد لا يحتاج جميع مستخدمي قاعدة المعطيات إلى الوصول إلى محتوى قاعدة المعطيات بأكمله (أو قد لا يُسمح لهم بذلك). ويُعتبر تعريف مجموعة المعطيات التي يسمح لمستخدم معين بالوصول إليها، وتحديد العمليات التي يستطيع إجرائها، من المهام الأساسية التي توكل إلى مدير قاعدة المعطيات كجزء من متطلبات أمن القاعدة.

8-2-4- التأمين والإصلاح

كأي جهاز إلكتروني أو ميكانيكي، تتعرض النظم الحاسوبية للأعطال. يمكن أن تنشأ هذه الأعطال عن عطل في القرص أو في التغذية أو في البرمجيات. وفي كل هذه الحالات

تتعرض معلومات قاعدة المعطيات للضياع. من مسؤوليات مدير قاعدة المعطيات اكتشاف هذه الأعطال وإعادة قاعدة المعطيات إلى الحالة التي كانت بها قبل العطل. ويتحقق هذا عادة من خلال عمليات التأمين وإجراءات الاسترجاع المتضمنة في قاعدة المعطيات.

8-2-5- التحكم بالوصول التزامن

عندما يقوم عدد من المستخدمين بتعديل قاعدة المعطيات بشكل متزامن فإن المعطيات يمكن أن تتعرض بعد فترة إلى تخريب وأن تصبح غير متجانسة. إن التحكم بتفاعل المستخدمين المتزامنين لقاعدة المعطيات هي إحدى مهام مدير قاعدة المعطيات.

نلاحظ بأن نظم قواعد المعطيات المصممة للاستثمار على حواسيب صغيرة (شخصية) لاتملك جميع المهام المذكورة سابقا. فبعضها يترك مهام التأمين والإصلاح والأمن للمستثمر، وبعضها يقصر استخدام الوصول إلى قاعدة لمستثمر واحد في الوقت نفسه.

8-3- مشغل قاعدة المعطيات Data base Administrator

من أحد الأسباب الرئيسية لوجود نظم إدارة قواعد المعطيات هو إيجاد تحكم مركزي لكل من المعطيات وبرمجيات الوصول لتلك المعطيات. يدعى الشخص المسؤول عن هذا التحكم المركزي مشغل قاعدة المعطيات (Data base Administrator D B A). تتضمن وظائف يدعى قاعدة المعطيات مايلي:

8-3-1- تعريف المخطط

يولد المخطط البدائي لقاعدة المعطيات بكتابة مجموعة من التعاريف التي تترجم من قبل مترجم لغة تعريف المعطيات (Data Definition Language) إلى مجموعة من الجداول المخزنة تخزيناً دائماً في قاموس المعطيات.

8-3-2- تعريف بنى التخزين طريقة الوصول

يقوم مدير النظام بتوليد بنى التخزين وطرق الوصول بكتابة مجموعة من التعاريف التي يترجمها مترجم لغة تعريف المعطيات إلى بنى تخزين المعطيات.

8-3-3- إجراء التعديلات على المخطط الفيزيائي

نادراً ما يضطر مشغل قاعدة المعطيات إلى إجراء لتعديلات على مخطط قاعدة المعطيات أو على وصف تنظيم التخزين الفيزيائي . تجري هذه التعديلات من خلال كتابة مجموعة من التعاريف التي يستخدمها مترجم لغات تعريف المعطيات ومترجم لغات تعريف وتخزين المعطيات في توليد تعديلات على الجداول الداخلية للنظام (مثلا : قاموس المعطيات).

8-3-4- تحديد سماحيات الوصول للمعطيات

ويُقصد بها توفير الأدوات التي تسمح لمدير قاعدة المعطيات بتنظيم أجزاء قاعدة المعطيات وتنظيم المستخدمين الذين يمكنهم الوصول إلى هذه القاعدة.

8-3-5- محددات شروط التكامل

تحفظ شروط التكامل في بنية خاصة يمكن قراءتها من قبل مدير قاعدة المعطيات عند حصول أي تعديل في النظام.

8-4- مستخدمو قاعدة المعطيات

الهدف الأساسي لنظم قواعد المعطيات هو إيجاد محيط للحصول على المعلومات من قاعدة المعطيات ولتخزين معلومات جديدة في قاعدة المعطيات. ثمة أربعة أنواع من مستخدمي قواعد المعطيات يتميزون من خلال طريقة تفاعلهم مع النظام.

8-4-1- مبرمجو التطبيقات

يتفاعل الحاسوب مع النظام من خلال أوامر تُكتب بلغة التعامل مع المعطيات DML التي تدخل بدورها في برامج مكتوبة بلغة تسمى "اللغة المضيقة" (مثل كوبول، PL/I، باسكال، C). وتدعى هذه البرامج البرامج التطبيقية. مثال: نظام المصارف يحوي برامج لتوليد الشبكات المدفوعة، حسابات الإبداع، حسابات الاسترداد، النقل بين الحسابات. وهكذا ...

تختلف لغة DML عن اللغة المضيقة من خلال الصيغة القواعدية. فعادة يجري وضع رموز خاصة قبل تعليمات DML تسمح لترجم اللغة المضيقة باستدعاء مترجم يحول جميع التعليمات المكتوبة بلغة معالجة المعطيات DML إلى طلب لإجراءات في اللغة المضيقة. وبالتالي ينتج لدينا برنامج مكتوب باللغة المضيقة يقوم مترجم هذه اللغة بتحويله إلى برنامج تنفيذي.

ثمة أنواع خاصة من لغات البرمجة تقوم بالجمع بين التحكم بالبنى في لغات البرمجة كما في لغة Pascal والتحكم بمعالجة أغراض قواعد المعطيات (مثل العلاقات). تدعى هذه اللغات لغات الجيل الرابع Fourth-generation Language وغالبا ما تحوي أدوات خاصة لتسهيل عملية توليد الاستمارات وإظهار المعطيات على الشاشة. تحوي أكثر نظم قواعد المعطيات التجارية لغات الجيل الرابع.

8-4-2- المستخدمون المحترفون

يتعاملون مع النظام بدون كتابة البرامج فهم يصيغون طلباتهم بلغة الاستفسار التي يوفرها نظام إدارة قواعد المعطيات مباشرة.

8-4-3- المستخدمون الاختصاصيون

هم بعض المستخدمون المتطورون الذين يقومون بكتابة تطبيقات خاصة لقواعد المعطيات والتي لا تدخل في المعالجة التقليدية للمعطيات. مثل تطبيقات التصميم بمساعدة الحاسوب، نظم المعرفة والنظم الخبيرة.

8-4-4- المستخدمون العاديون

هم مستخدمون يتفاعلون مع النظام من خلال أحد البرامج التطبيقية المكتوبة سابقاً.

8-5- مدير الملفات

هو أحد أجزاء نظام التشغيل. يُعتبر هذا البرنامج مسؤولاً عن حجز مناطق التخزين على قرص التخزين وإدارة بنية المعطيات المستخدمة لتمثيل المعلومات المخزنة على القرص.

8-6- معالج الاستفسار

يقوم هذا البرنامج بترجمة التعليمات المكتوبة بلغة الاستفسار إلى مجموعة من التعليمات في المستوى الأدنى التي يفهمها مدير قاعدة المعطيات. كما أنه يقوم بتحويل طلبات المستخدم إلى استفسارات مكافئة أكثر فعالية لإيجاد طريقة جيدة لتنفيذ الاستفسار.

8-7- المترجم الأولي لتعليمات لغة التعامل مع المعطيات

DML pre-compiler

يقوم بتحويل تعليمات لغة التعامل مع المعطيات DML المتضمنة في البرامج التطبيقية إلى صيغة مكافئة تستخدم تعليمات اللغة المضافة.

8-8- مترجم لغة تعريف المعطيات DDL compiler

يقوم بتحويل تعليمات DDL إلى مجموعة من الجداول التي تتضمن وصفاً لبنية قاعدة المعطيات (metadata).

بالإضافة إلى ما سبق فإن العديد من بنى المعطيات الضرورية لتحقيق نظام إدارة قواعد المعطيات ، أهمها:

ملفات المعطيات Data files حيث يجري تخزين قاعدة المعطيات.

قاموس المعطيات Data dictionary حيث تخزن المعطيات حول بنية قاعدة المعطيات.

الفهارس وتستخدم لتسريع الوصول إلى المعطيات.

تمارين الفصل الأول

- 1- اذكر أربعة فروق أساسية بين نظم إدارة الملفات ونظم إدارة قواعد المعطيات
- 2- اشرح الفرق بين ارتباط المعطيات المنطقي والفيزيائي
- 3- تقوم شركة الطيران العربية السورية بتنظيم رحلات جوية تنطلق من عدد من المدن السورية (دمشق، حلب، اللاذقية، دير الزور) إلى عواصم ومدن عديدة. تمتلك الشركة أسطولاً جويّاً يضم عدداً من طائرات نقل الركاب لكل منها سعة محددة (العدد الأعظمي للركاب). ويجري تكليف كل طائرة بعدد من الرحلات. يحجز الزبائن الأماكن قبل موعد الرحلة بثلاثة أيام على الأقل، ويجري تسجيل أسماء وعناوين الركاب في قائمة خاصة بكل رحلة.
- أوجد المخطط المنطقي باستخدام النموذج الهرمي
- بكم طريقة يمكن إعطاء هذا المخطط
- كيف يتم إختيار المخطط المناسب
- أعط مثلاً عن محتوى القاعدة وفق مخطط مختار
- 4- لتكن قاعدة المعطيات المذكورة في التمرين السابق
- أوجد المخطط المنطقي للقاعدة باعتماد النموذج الشبكي
- أعط مثلاً عن محتوى القاعدة وفق هذا المخطط.

الفصل الثاني

المخطط المفاهيمي لقاعدة المعطيات

نموذج كيانات-ارتباطات

مقدمة

رأينا في الفصل السابق أن المبدأ العام لتصميم قاعدة المعطيات يعتمد مبدأ التجريد المتتالي للواقع المدروس، بحيث يجري في كل مرحلة وصف محتوى قاعدة المعطيات بشيء من التفصيل الإضافي عن المرحلة التي سبقتها.

ورأينا أن لهذا المنحى فوائد عديدة أهمها عزل المعطيات عن محيط العمل وعن البرامج التي تتعامل مع المعطيات.

إن نقطة البداية في إنشاء أي قاعدة معطيات هي إنشاء المخطط المفاهيمي Conceptual shcema .

جرى في نهاية الستينيات وبداية السبعينيات من القرن العشرين تطوير عدة نماذج وطرق لإنشاء المخططات المفاهيمية. وقد كانت طريقة مخططات الكيانات والارتباطات : ERD Entity Relationship Diagrams التي اقترحها Chen ، والتي خضعت للعديد من عمليات التطوير والتعديل، أشهر هذه الطرق، وهذا ما جعلها تطفئ على غيرها من الطرق وتصبح معتمدة في معظم الأنظمة المساعدة في هندسة البرمجيات CASE.

يعتمد نموذج المعطيات كيان-ارتباط على تمثيل العالم الحقيقي بمجموعة من الأغراض تُسمى كيانات، وتعريف الارتباطات فيما بينها. طُوّر هذا النموذج لتسهيل تصميم قواعد المعطيات، فهو يسمح بتحديد مخطط المؤسسة الذي يمثل البنية المنطقية لقاعدة المعطيات.

1- الكيانات وصفوف الكيانات

الكيان : هو غرض مميز عن غيره من الأغراض التي سيجري تخزينها في قاعدة المعطيات.

فمثلاً الطالب سعيد الذي يحمل البطاقة الجامعية ذات الرقم 402727 الصادرة عن جامعة دمشق هو كيان لأنه لا يوجد شخص آخر في العالم يحقق هذه المواصفات.

نلاحظ من المثال السابق أن الكيانات يمكن أن تكون مادية concrete أو محسوسة (شخص، كتاب، سيارة، قاعة، ... إلخ) كما يمكن أن تعبر عن أشياء مجردة abstract (يوم السبت الواقع في 2000/7/15، حجز مقعد في رحلة الطائرة السورية المغادرة إلى لندن يوم الأحد في 2000/7/16، ... إلخ).

صفوف الكيانات : بسبب العدد الكبير الذي يمكن أن تحويه قاعدة المعطيات، يجري تجميع الكيانات المتشابهة في مجموعات تسمى صفوف الكيانات. مثل : صف الأشخاص الذين لهم حساب في مصرف، صف الحسابات المصرفية، صف الموظفين، صف السيارات، صف القروض،

من الممكن أن تتقاطع صفوف الكيانات. مثال : يمكننا تعريف ضمن مؤسسة مصرفية صفوف الكيانات التالية :

• صف موظفي المصرف.

- صف الزبائن (الأشخاص الذين لديهم حساب مصرفي في المصرف).
- صف الأشخاص الذين لهم علاقة في المصرف، والذين يمكن أن ينتموا إلى صف الموظفين أو صف الزبائن أو لا ينتموا إلى أي من الصنفين السابقين.

طريقة تمثيل الكيان : يُمثل الكيان بمجموعة من الواصفات. ولكل واصف مجموعة من القيم الممكنة. تُسمى مجموعة القيم الممكنة لواصفة بمجال الوصف Attribute Domain، من ثم يتصف الكيان بأزواج من الشكل (واصف، قيمة الوصف).

مثال :

يتصف زبون مصرف بالواصفات التالية : (الاسم، رقم الهوية، المدينة، الشارع) ويتصف زبون معين يمثل كياناً ب :

{ (الاسم: محمد)، (رقم الهوية: 1213141)، (المدينة: دمشق)، (الشارع: المنارة) }

يشبه مفهوم صف الكيانات مفهوم نمط المعطيات في لغات البرمجة، حيث يجري تعريف قالب عام يمثل مجموعة من الأغراض، يجري بواسطتها تعريف متحويلات لكل منها قيمة معينة، ولكنها تشترك في البنية، إذ يوافق مفهوم المتحول في لغات البرمجة مفهوم الكيان في نموذج الكيانات والارتباطات.

تضم قاعدة المعطيات مجموعة من صفوف الكيانات يحوي كل منها عدداً غير محدود من الكيانات.

يبين الشكل (1) جزءاً من قاعدة معطيات المؤسسة المصرفية، يبين صف الكيانات: الزبائن والحسابات المصرفية. يبين هذا الشكل جزءاً من محتوى قاعدة المعطيات من خلال محتوى صف الكيانات السابقين :

الاسم	رقم الهوية	المدينة	الشارع	رقم الحساب	رصيد الحساب
محمد	1213141	دمشق	المنارة	120	1234
محمود	132441	دمشق	قصاع	133	234
يوسف	17771	حمّاة	النواعير	556	144434
سامي	5558881	حمص	الدبلان	16	123499
وسام	667789	دمشق	الميدان	188	8234
حسام	6678986	اللاذقية	الكورنيش	6789	33456
				3356	88902

الزبائن

الحسابات المصرفية

الشكل (1) : مثال لمحتوى صفحي الكيانات

2- الارتباط وصفوف الارتباطات

الارتباط : هو علاقة تربط مجموعة من الكيانات بعضها ببعض. فمثلاً يمكن أن نجد ارتباطاً بين الشخص محمد والحساب المصرفي ذي الرقم 133. يشير هذا الارتباط إلى أن محمداً هو زبون للمصرف وله حساب مصرفي رقمه 133.

صف الارتباطات : هو مجموعة من الارتباطات من نوع واحد. يمكننا التعبير رياضياً عن صف الارتباطات كما يلي :

إذا كانت $E1, E2, E3, \dots, En$ مجموعة من صفوف الكيانات فيعرف صف الارتباطات R كمجموعة جزئية من الجداء الديكارتي

$$\{ (e1, e2, e3, \dots, en) \mid e1 \in E1, e2 \in E2, e3 \in E3, \dots, en \in En \}$$

حيث $(e1, e2, e3, \dots, en)$ هي علاقة ارتباط.

مثال : يبين الشكل (2) علاقة ارتباط بين صفي الكيانات : الزبائن. الحسابات المصرفية ، ونسمي هذه العلاقة بـ زبون-حساب.

الشارع	المدينة	الرقم	الاسم	رقم الحساب	رصيد الحساب
المنارة	دمشق	1213141	محمد	120	1234
قصاع	دمشق	132441	محمود	133	234
النواعير	حماة	17771	يوسف	556	14443 4
الدبلان	حمص	5558881	سامي	16	12349 9
الميدان	دمشق	667789	وسام	188	8234
الكورنيش	اللاذقية	6678986	حسام	6789	33456
				3356	88902

الحسابات المصرفية العلاقة الزبائن

زبون-حساب

الشكل (2) : علاقة ارتباط

العلاقة السابقة هي علاقة ثنائية بين صفين من صفوف الكيانات. ويلاحظ أن أغلب علاقات الارتباط في قواعد المعطيات هي علاقات ارتباط ثنائية، ويسمى مصممو قواعد المعطيات لتحويل علاقات الارتباط غير الثنائية إلى مجموعة من علاقات ارتباط ثنائية، لأن هذا النوع من الارتباطات هو الأكثر فهماً وقرباً للواقع.

يمكن أن تتصف علاقة الارتباط بمجموعة من الواصفات. مثال : ربط الواصف "تاريخ" بعلاقة الارتباط زبون-حساب. يحدد هذا الواصف تاريخ الحالة التي أخذ فيها حساب الزبون المصرفي.

3- الواصفات

إن مفهومي صف الكيانات وصف الارتباطات غير كافيين وحدهما لتحديد مخطط قاعدة المعطيات. فيمكننا الربط بين صفي كيانات باستخدام صف ارتباطات بطرق مختلفة، ويظهر الفرق الرئيسي في طريقة معالجة الواصفات المرتبطة بكل من الكيانات والارتباطات.

فإذا أردنا مثلاً تعريف قاعدة معطيات تتضمن معلومات عديدة من بينها معلومات عن الموظفين والهواتف التي يملكونها. يمكننا تعريف مخطط كيانات-ارتباط للقاعدة بطريقتين:

الطريقة الأولى : تعريف صف كيانات واحد نسميه "موظفاً" ونربط به الواصفات التالية :

• اسم الموظف

• رقم الهاتف

الطريقة الثانية : تعريف صفي كيانات (موظف، هاتف) وربطهما بصف الارتباطات (موظف-هاتف). وفي هذه الحالة يكون لدينا ما يلي :

• الواصفات المرتبطة بصف الكيانات "موظف" هي : اسم الموظف

• الواصفات المرتبطة بصف الكيانات "هاتف" هي : رقم الهاتف، مكان

التركيب

• علاقة الارتباط موظف-هاتف التي تعبر عن العلاقة بين الموظفين والهواتف التي يملكونها.

نلاحظ أن الفرق بين التعريف بالطريقة الأولى والطريقة الثانية هو أن التعريف الأول يحدد لكل موظف رقم هاتف واحد فقط، أما التعريف الثاني فيسمح بأن يكون للموظف أكثر من هاتف أو ألا يملك الموظف أي هاتف. إن التعريف بالطريقة الثانية أعم ويعبر تعبيراً أدق عن الواقع الحقيقي.

ومن هنا نلاحظ أن هناك سؤالاً دائماً يطرح على مصممي قاعدة المعطيات : ما هي الكيانات التي يجب أن تعتبر واصفات؟ وماهي الكيانات التي يجب أن تُعتبر صفوف كيانات؟

الجواب ليس بسيطاً ويعتمد على بنية المسألة المراد نمذجتها، والمعنى المرتبط بالواصفات في السؤال المطروح.

4- تمثيل الشروط

يُشترط في مخطط الكيانات والارتباط أن يُعبر عن الشروط التي يجب أن يحققها محتوى قاعدة المعطيات في كل لحظة. إن أحد أهم هذه الشروط هو درجة الارتباط، وهو مقدار يُعبر عن الكيانات التي يمكن أن ترتبط بكيانات أخرى عبر مجموعة الارتباطات.

تُستخدم درجة الارتباط غالباً في وصف الارتباطات الثنائية، وتدخل أحياناً في وصف علاقات الارتباط بين أكثر من صفي كيانات. سنهتم حالياً بالارتباطات الثنائية.

لتكن R علاقة ارتباط ثنائية بين صفي الكيانات A , B . تأخذ درجة الارتباط إحدى الحالات التالية:

واحد-واحد : كل كيان من صف الكيانات A يرتبط على الأكثر بكيان واحد من صف الكيانات B ، وبالعكس كل كيان من صف الكيانات B يرتبط على الأكثر بكيان واحد من صف الكيانات A.

واحد-كثير : من الممكن لكيان من A أن يرتبط بأي عدد من الكيانات في B ، وكل كيان من B يرتبط على الأكثر بكيان واحد من A.

كثير-واحد : كل كيان من A يرتبط على الأكثر بكيان من B ويمكن لكيان من B أن يرتبط بعدد من الكيانات من A.

كثير-كثير : يمكن لكيان من A أن يرتبط بعدد من الكيانات من B ، وبالعكس يمكن لكيان من B أن يرتبط بعدد من الكيانات من A.

إن تحديد درجة الارتباط يعتمد على ملاحظة الواقع الذي تجري نمذجته بواسطة مجموعات الارتباطات. فإذا نظرنا إلى الارتباط بين صف الكيانات "زبون" وصف الكيانات "حساب" فإن هذا الارتباط يبدو للوهلة الأولى أنه من نوع واحد-واحد ولكن قد يتبين من مناقشة العاملين في المصرف أنه يُسمح لزبون معين أن يمتلك أكثر من حساب مصرفي. في هذه الحالة يتحول الارتباط زبون-حساب إلى النوع واحد-كثير، وفي مرحلة تالية يمكن أن تستنتج أن المصرف يسمح بفتح حسابات مشتركة (لأفراد العائلة الواحدة مثلاً) وعندها يتحول الارتباط إلى نوع كثير-كثير.

5- المفاتيح

من المهم أن نحدد كيف نميز الكيانات التي تنتمي إلى صف كيانات معين، وكذلك الارتباطات التي تنتمي إلى صف ارتباطات معين. ومن وجهة نظر تصميمية، يجري التمييز بين الكيانات في صف واحد بواسطة الواصفات.

5-1- المفتاح الأعلى أو المفتاح الرئيسي

يُستخدم مفهوم المفتاح الأعلى Superkey للتمييز بين الكيانات والروابط بعضها عن بعض. ويجري هذا التمييز بواسطة قيم الواصفات المرتبطة بها. ويعرف بأنه مجموعة تضم واصفاً أو أكثر تسمح مجتمعة بتمييز كيان واحد من مجموعة من الكيانات المنتمية إلى صف واحد.

أمثلة :

– (الرقم الذاتي) يُكوّن هذا الوصف مفتاحاً رئيسياً لصف الكيانات (الزبائن) وذلك لأنه قادر على تمييز كيان : زبون من زبون آخر. فالقيمة 12345 للرقم الذاتي تُميز زبوناً وحيداً من صف الزبائن.

– (الرقم الذاتي، اسم الزبون) هذه مجموعة من الواصفات تُكوّن مفتاحاً رئيسياً لأنها قادرة على تمييز كيان : زبون من زبون آخر، ولكن (اسم الزبون) كواصف لا يُكوّن وحده مفتاحاً رئيسياً لأنه يمكن لعدة أشخاص أن يحملوا الاسم نفسه. فالاسم غير كافٍ وحده لتمييز كيان عن آخر.

رأينا في المثال السابق أن المفتاح الرئيسي لصف كيانات أو ارتباطات يمكن أن يحوي عدداً غير محدود من الواصفات. فإذا كانت K مجموعة من الواصفات التي تُكوّن

مفتاحاً رئيسياً، فإن أي مجموعة من الواصفات تحوي المجموعة K هي مفتاح رئيسي أيضاً.

غالباً ما نهتم بالمفتاح المرشح **Candidate key** وهو مفتاح رئيسي مؤلف من مجموعة من الواصفات ولا توجد مجموعة جزئية من هذه المجموعة تُكوّن مفتاحاً رئيسياً.

5-2 - المفتاح الأولي Primary key

هو مفتاح مرشح اختاره مصمم قاعدة المعطيات كطريقة أساسية لتمييز الكيانات عن بعضها البعض والمنتمية إلى صف كيانات واحد.

يُوصف صف الكيانات بأنه صف كيانات ضعيف إذا لم يحو مجموعة من الواصفات الكافية لتكوّن مفتاحاً أولياً، ويُوصف بأنه صف كيانات قوي في الحالة العاكسة.

رأينا سابقاً أن المفتاح الأولي لمجموعة كيانات يميز الكيانات بعضها عن بعض. ونحتاج إلى تقنية مشابهة لتمييز الارتباطات المنتمية لصف الارتباطات.

سنصنف أولاً علاقة الارتباط، ثم نعرف المفتاح الأولي لعلاقة الارتباط هذه.

5-3 - وصف علاقة الارتباط

ليكن لدينا العلاقة R علاقة الارتباط بين مجموعة من صفوف الكيانات $E_1, E_2, E_3, \dots, E_n$ ولنرمز بـ $PK(E_i)$ إلى المفتاح الأولي لصف الكيانات E_i . ولنفتراض أن أسماء الواصفات المميزة لجميع المفاتيح الأولية وحيدة (أي لا يوجد أسماء واصفات مشتركة بين $PK(E_1), PK(E_2), \dots, PK(E_n)$) وإن وجدت فإننا نغير أسماء الواصفات.

ولنفترض أن علاقة الارتباط R لا تحوي أي واصف. فإن واصفات علاقة الارتباط R ولنرمز إليها بـ $ATTRIBUTE(R)$ هي .

$$PK(E1) \cup PK(E2) \cup \dots \cup PK(En)$$

إذا كان R مجموعة من الواصفات ولتكن $\{a1, a2, a3, \dots, am\}$ فإن واصفات علاقة الارتباط R ولنرمز إليها بـ $ATTRIBUTE(R)$ هي :

$$PK(E1) \cup PK(E2) \cup \dots \cup PK(En) \cup \{a1, a2, a3, \dots, am\}$$

مثال : لتكن لدينا علاقة الارتباط زبون-حساب مصرفي التي تربط بين صفي الكيانات : زبائن، حسابات مصرفية. وليكن المفتاح الأولي لصف الزبائن هو " رقم الزبون"، والمفتاح الأولي لصف الحسابات المصرفية هو " رقم الحساب المصرفي"، وليكن لعلاقة الارتباط زبون-حساب مصرفي واصف هو "التاريخ".

الواصفات التي تميز علاقة الارتباط زبون-حساب مصرفي هي :

$$ATTRIBUTE(R) = \{ client-nb \} \cup \{ account-nb \} \cup \{ date \}$$

5-4- المفتاح الأولي لعلاقة الارتباط

يتعلق المفتاح الأولي لعلاقة الارتباط R بالواصفات المرتبطة بها وبدرجة الارتباط بصفوف الكيانات الموجودة على حدي العلاقة.

فإذا لم توجد واصفات مرتبطة بالعلاقة R ، فإن مجموعة الواصفات المؤلفة لـ $ATTRIBUTE(R)$ تكون مفتاحاً رئيسياً للعلاقة R . ويكون هذا المفتاح المفتاح الأولي للعلاقة R إذا كانت حدود العلاقة من الشكل كثير-كثير، أما إذا كانت العلاقة من الشكل كثير-واحد فإن المفتاح الأولي للعلاقة يتألف من المفتاح الأولي لصف الكيانات الذي يرتبط بالكثير.

وفي حال وجود واصفات مرتبطة بالعلاقة R فيتكون المفتاح الأولي لها كما أوردنا سابقاً، مضافاً إليه واصف أو أكثر من واصفات R .

مثال : لنأخذ صفّي الكيانات "زبون" و "مصرف" وعلاقة الارتباط "زبون-مصرف" التي تربط كل زبون بالمصارف التي يتعامل معها. ولنفتراض أن لعلاقة الارتباط هذه واصفة هي (نوع) ، تصف طبيعة علاقة الارتباط بين الزبون والمصرف.

فإذا كان بإمكان مصرف أن يؤدي دورين مختلفين في علاقته مع زبون معين، فإن المفتاح الأولي لعلاقة الارتباط "زبون-مصرف" تتألف من اجتماع المفاتيح الأولية المتعلقة بـ "زبون" و "مصرف" مضافاً إليها الوصفة (نوع).

وفي حال وجود علاقة من نوع واحد فقط بين "مصرف" و "زبون"، فإن الوصفة (نوع) لا تُكوّن جزءاً من المفتاح الأولي لعلاقة الارتباط "زبون-مصرف"، والمفتاح الأولي هو فقط اجتماع المفاتيح الأولية لكل من "زبون" و "مصرف".

5-5- صفوف الكيانات الضعيفة

نقول عن صف كيانات إنّه صف كيانات ضعيف إذا كان لا يحوي مفتاحاً أولياً.

مثال :

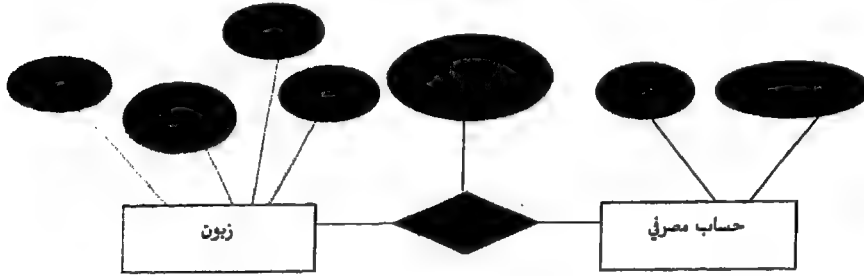
ضمن قاعدة معطيات تحوي معلومات عن القروض وعمليات سداد هذه القروض بواسطة مجموعة من الدفعات، يجري تحديد صفّي كيانات : الكيانات "دفع" و صف كيانات "قرض". لنأخذ صف الكيانات "دفع" Payment الذي لديه ثلاث واصفات (رقم الدفع، تاريخ الدفع، الكمية) نلاحظ أن عملية الدفع تتعلق بالقرض المأخوذ ويمكن أن يوجد كيانات غير متمايزين في صف الكيانات "دفع" لقرضين متمايزين. من ثمّ لا يوجد مفتاح أولي لصف الكيانات "دفع" وهو صف كيانات ضعيف.

يجري تكوين المفتاح الأولي لصف كيانات ضعيف من المفتاح الأولي لصف الكيانات القوي المرتبط وجوده به مضافاً إليه مجموعة الواصفات المميزة لصف الكيانات الضعيف. في مثالنا السابق يكون المفتاح الأولي (رقم القرض، رقم الدفع).

6- مخطط تمثيل كيان-ارتباط E-R Diagram

يمكن التعبير عن بنية قاعدة المعطيات بيانياً باستخدام مخطط كيان-ارتباط. يتألف هذا المخطط من المكونات التالية :

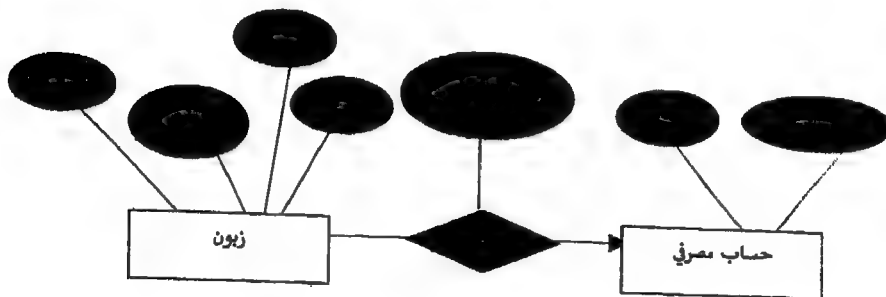
- مستطيلات : تُمثل صفوف الكيانات،
- مستطيل مضاعف : لتمثيل صف الكيانات الضعيف.
- قطوع : تُمثل الواصفات،
- معينات : تمثل صفوف الارتباطات،
- خطوط : تربط بين صفوف الكيانات بالواصفات، وتربط صفوف الكيانات بصفوف الارتباطات.
- خط تحت الواصفات المميزة لصف الكيانات.
- خط متقطع تحت الواصفات المميزة لصف الكيانات الضعيف.
- معيّن مضاعف لتمثيل علاقة ارتباط صف كيانات ضعيف بصف كيانات قوي.



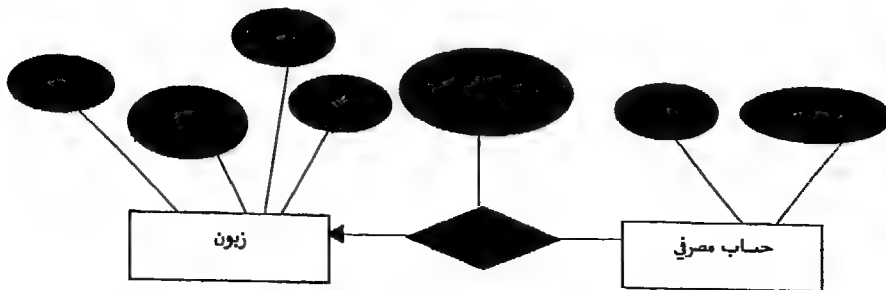
الشكل (3) : مثال لمخطط كيانات -ارتباطات

لنتأمل الشكل (3). المخطط E-R الذي يتألف من صفي كيانات هما : زبون. حساب مصرفي. المرتبطان بعلاقة ارتباط ثنائية هي زبون-حساب مصرفي، الواصفات المرتبطة بالزبون هي : اسم الزبون، رقم الهوية، الشارع، المدينة. والواصفات المرتبطة بالحساب المصرفي هي : رصيد، رقم حساب. أما الواصفات المرتبطة بالعلاقة زبون-حساب فهي : التاريخ (تاريخ فتح الحساب).

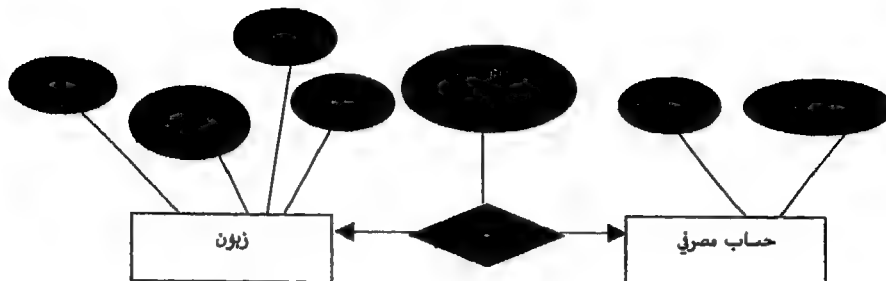
يمكن لعلاقة الارتباط زبون-حساب أن تكون من الشكل كثير-كثير، واحد-كثير، كثير-واحد، أو أخيراً واحد-واحد. للتمييز بين هذه الأنواع سوف نستخدم خطاً موجهاً (→) أو خطاً غير موجه من الشكل (—) للربط بين صف الارتباطات وصف الكيانات المعني. يُستخدم الخط الموجه من صف الارتباطات "زبون-حساب" إلى صف الكيانات "حساب مصرفي" لبيان أنه يرتبط بصف الكيانات "زبون" بعلاقة من الشكل واحد-واحد أو كثير-واحد ، ولا يمكنه أن يرتبط بعلاقة كثير-كثير أو واحد-كثير بصف الكيانات "زبون".



(a)



(b)



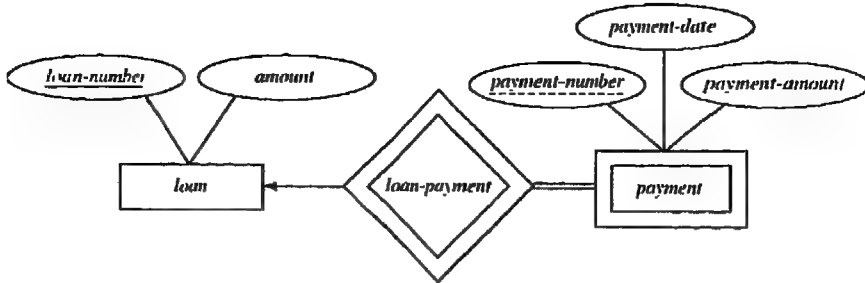
(c)

الشكل (4) : مثال لمخطط كيانات - ارتباطات

أما الخط غير الموجه الذي يربط صف الارتباطات "زبون-حساب" بصف الكيانات "حساب" فيبين بأنه يرتبط بعلاقة كثير-كثير أو كثير-واحد بصف الكيانات زبون

يبين الشكل (3) أن علاقة الارتباط "زبون-حساب" هي من النوع كثير-كثير. يبين الشكل (4) الحالات الثلاث : (a) علاقة الارتباط "زبون-حساب" هي من النوع واحد-كثير من الزبون إلى الحساب ، (b) علاقة الارتباط "زبون-حساب" هي من النوع كثير-واحد من الزبون إلى الحساب ، (c) علاقة ارتباط من النوع واحد-واحد بين الزبون والحساب حيث يكون الخطان اللذان يربطان صف الارتباط بكل من صفي الكيانات موجهين.

يُمثل الشكل (5) علاقة ارتباط ضعيفة بين صف الكيانات الضعيف "دفع" وصف الكيانات القوي "قرض".



الشكل (5) : تمثيل مخطط كيانات-ارتباط

7- نموذج كيان-ارتباط موسّع

يمكن للمفاهيم الأساسية في نموذج كيان-ارتباط أن تُوفر معظم الوظائف المطلوبة لنمذجة قاعدة المعطيات. وتُصبح المفاهيم الأساسية في قواعد المعطيات أكثر وضوحاً بإضافة بعض التوسع إلى النموذج الأساسي لنموذج كيان-ارتباط. يتضمن هذا التوسع علاقات التخصيص والتعميم (مستوى أعلى، مستوى أدنى)، وتوريث الواصفات، والتجميع.

7-1- علاقة التخصيص

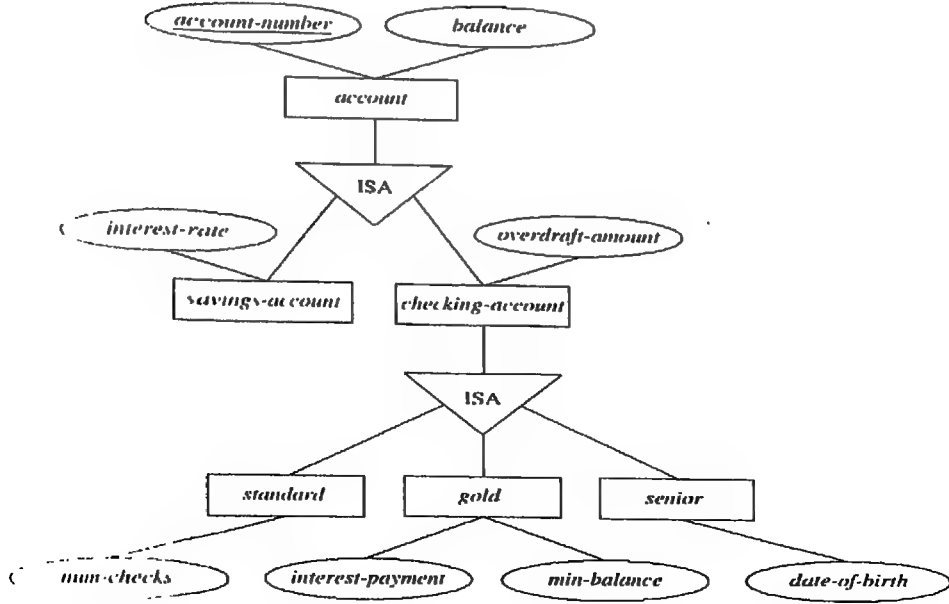
يمكن أن يوجد ضمن مجموعة كيانات، كيانات مميزة عن الكيانات الأخرى بطريقة ما. إذ يمكن أن تمتلك مجموعة جزئية من الكيانات واصفات غير مشتركة مع بقية الكيانات في المجموعة. إن نموذج E-R الموسع يقدم طرقاً لتمثيل هذا التمايز بين تجمع الكيانات، وتسمى عملية تحديد مجموعات جزئية من مجموعة الكيانات الأساسية عملية تخصيص. مثال : لناخذ مجموعة الكيانات "الحساب" والموصفة بالواصفات : رصيد الحساب، المبلغ

صُنف الحساب فيما بعد إلى حسابين : حساب توفير وحساب شيكات. يمتلك كل نوع من الحسابين السابقين كل واصفات الحساب الأساسية (رقم الحساب، الرصيد)، ويحوي كل منهما واصفات إضافية، ففي حالة حساب التوفير يضاف إلى هذين الواصفين واصف جديد يتعلق بمعدل الفائدة، وفي حالة حساب الشيكات يضاف واصف جديد يتعلق بالرصيد الأدنى للحساب. إن عملية التمييز بين النوعين السابقين للحساب المصرفي هي عملية تخصيص.

يمكن لعملية التخصيص أن تجري بعدة هيئات، ففي مثالنا السابق كان التخصيص حسب نوع الحساب، ومن الممكن أن يكون حسب مالك الحساب. عندها تكون النتيجة (حساب تجاري، وحساب شخصي).

يجري تمثيل التخصيص في مخطط الكيانات والارتباطات بواسطة مثلث يوضع بداخله كلمة "ISA" دلالة على "is a". تُمثل هذه العلاقة علاقة من نوع "Superclass-Subclass" في النمذجة الغرضية التوجه.

يُمثل المخطط التالي الشكل (6) علاقة التخصيص للكيان "حساب" إلى "حساب توفير" و"حساب شيكات" وعلاقة تخصيص للكيان "حساب شيكات" إلى "حساب عادي". "حساب مجمد"، "حساب فعال"، حيث "حساب عادي" هو حساب للأشخاص المتقدمين في العمر، "حساب مجمد" هو حساب برصيد لا يقل عن حد معين ويتقاضى الزبون عليه فائدة، أما "حساب فعال" فهو الحساب الجاري الذي يتعامل به الزبون باستخدام عدد من الشيكات.



الشكل (6) : مثال لعلاقة تخصيص

7-2- علاقة التعميم

إن عملية التعميم هي عملية تصميم من الأسفل إلى الأعلى (bottom-up) يجري فيها مجموعات الكيانات الجزئية وذلك اعتماداً على الخواص المشتركة في مجموعة كيانات أشمل.

مثال : من الكيان "حساب الشيكات" الموصف بالواصفات : رقم الحساب. الرصيد. الرصيد الأدنى. وكيان "حساب التوفير" الموصف بالواصفات : رقم الحساب. المبلغ. معدل الفائدة.

بالتعميم نجد كيان " الحساب " في المستوى الأعلى المرتبط بـ "حساب الشيكات" و"حساب التوفير" في المستوى الأدنى.

كما نرى أن التعميم هو عكس التخصيص، ولذلك فإننا لا نميز بينهما في مخطط الكيانات والارتباطات.

7-2-1- توريث الواصفات

الخاصة المميزة للكيانات في المستوى الأعلى والأدنى المولدة بالتخصيص والتعميم هي توريث الواصفات. وبوجه عام يطبق على أي جزء من المخطط E-R و يمثل تخصيصا أو تعميما ما يلي :

الواصفات والعلاقات المعرفة في المستوى الأعلى تطبق على جميع الكيانات في المستويات الدنيا المرتبطة به.

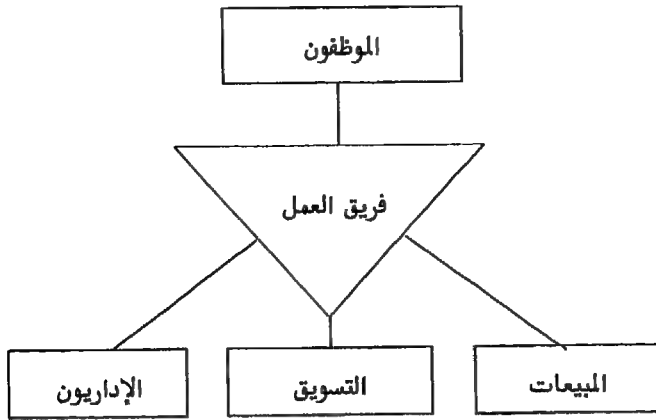
تتعلق الواصفات المعرفة على الكيانات في المستوى الأدنى فقط بالكيان المعني في المستوى الأدنى.

7-2-2- القيود التصميمية

لنمذجة مؤسسة نمذجة صحيحة يحتاج مصمم قواعد المعطيات إلى وضع شروط على تعميم محدد. تحدد هذه الشروط الكيانات التي تكون مجموعة كيانات مستوى أدنى، ويجري ذلك بطريقتين :

تعريف سابق Predefined Condition: يجري تحديد الأعضاء بواسطة شرط سابق التعريف. مثال : في مجموعة الكيانات "حساب" يجري تحديد مجموعتين للكيانات في المستوى الأدنى (حساب توفير، حساب شيكات) حسب الوصف " نوع الحساب".

تعريف ديناميكي (من قبل المستثمر) User defined : حيث يجري تحديد أعضاء مجموعة كيانات من قبل مستخدم قاعدة المعطيات. مثال : لنفترض أنه يجري توزيع موظفي المصرف على فرق عمل بعد أن يكونوا قد أمضوا ثلاثة أشهر من العمل. لا تجري عملية التوزيع آليا وإنما يجريها مستثمر النظام. إن مجموعات الكيانات المثلة لفرق العمل تكون المستوى الأدنى (التخصيص) لمجموعة كيانات "موظفو المصرف".



ترتبط عملية التعميم/التخصيص بمجموعات لها الخصائص التالية :

منفصلة Disjoint : أي كيان من المستوى الأدنى (ضمن مجموعات التخصيص) يرتبط بكيان واحد على الأكثر من مجموعة كيانات المستوى الأعلى. مثال : "الحساب" ومجموعتا "حساب الشيكات" و "حساب التوفير".

التراكب Overlapping : يمكن لكيان من المستوى الأعلى (مجموعة التعميم) أن يرتبط بأكثر من كيان من المستوى الأدنى (مجموعات التخصيص). مثال : "فرق العمل" و"موظفو المصرف" إذ يمكن أن يشترك موظف بأكثر من فريق عمل .

ويمكن لعملية التعميم/التخصيص أن تكون :

كلية Total : أي إن أي كيان من المستوى الأعلى يرتبط على الأقل بكيان من المستوى الأدنى.

جزئية : يمكن أن يوجد كيان في المستوى الأعلى لا يرتبط بأي كيان في المستوى الأدنى.

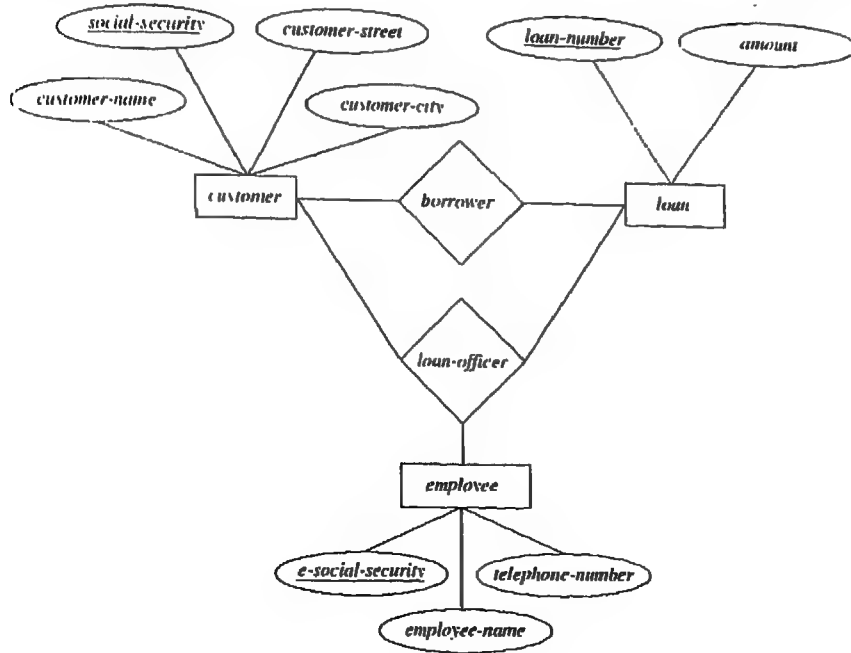
7-3 التجميع Aggregation

هو اعتبار مخطط جزئي من المخطط E-R صف كيانات يمكن استخدامه في ارتباطات أخرى، وهذا ما يسمح لنا بمعالجة صف الكيانات المجمع كوحدة مستقلة، دون النظر إلى تفاصيل بنيتها الداخلية.

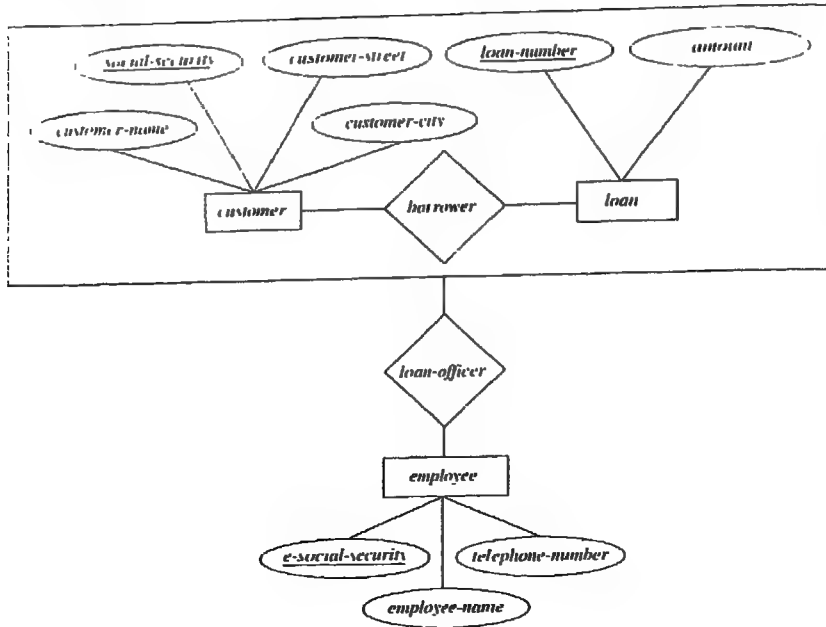
مثال : لנأخذ قاعدة المعطيات التي تحوي معلومات عن الزبائن المتعاملين مع المصرف وقروضهم، ولنفتراض أنه من الممكن وجود موظف في المصرف مسؤول عن زوج المعلومات (زبون، قرض).

أفضل طريقة لتمثيل المعلومات السابقة هي باستخدام التجميع للعلاقة "يقترض" بين صفي الكيانات (زبون، قرض) في صف للكيانات لنسمه "اقتراض".

يبين الشكل التالي المخطط E-R قبل وبعد استخدام فكرة التجميع.



الشكل (7) : مخطط قاعدة المعلومات قبل استخدام فكرة التجميع



الشكل (8) : مخطط قاعدة المعطيات بعد استخدام فكرة التجميع

8- مراحل التصميم في نموذج كيانات-ارتباطات

إن نموذج المعطيات العالي المستوى يخدم مصمم قاعدة المعطيات بتقديم العمل التصميمي الواجب تحديده، وفي الأحوال العادية- متطلبات مستخدمي قاعدة المعطيات، والبنية الواجب توفرها في قاعدة المعطيات لتحقيق هذه المتطلبات. ومن ثم المرحلة الأولى في تصميم قاعدة المعطيات هي تحديد المعطيات التي يحتاج إليها المستخدمون المستقبليون لقاعدة المعطيات. خرج هذه المرحلة هو تحديد متطلبات المستخدم.

المرحلة الثانية : يقوم المصمم باختيار نموذج المعطيات وتطبيقه لترجمة هذه المتطلبات إلى المخطط التصميمي لقاعدة المعطيات . المخطط الناتج يعطي منظرا مفصلا للمؤسسة. ولأننا درسنا فقط النموذج كيان-ارتباط فإننا سوف نستخدمه لتطوير المخطط التصميمي. وضمن هذا المخطط سوف يجري تحديد جميع الكيانات والارتباطات والواصفات وخريطة الشروط. وسوف يقوم المصمم بمراجعة المخطط ليتوثق من تحقيقه لجميع المتطلبات ومن عدم وجود أي تعارض فيما بينها، ويقوم بحذف التكرار إن وجد.

سوف يشير المخطط التصميمي الكامل إلى المتطلبات الوظيفية للمؤسسة. يقوم المستخدم في هذه المرحلة (وصف المتطلبات الوظيفية) بوصف أنواع العمليات (أو التحويلات) التي سوف تجرى على المعطيات. مثال : حذف، إضافة، تعديل ، أو بحث للمعطيات.

إن عملية الانتقال من نموذج المعطيات المجرد إلى تنفيذ قاعدة المعطيات تنفذ بمرحلتين تصميم نهائيتين. في مرحلة التصميم المنطقي يجري نقل المخطط التصميمي العالي المستوى إلى نموذج المعطيات المنفذ لنظام إدارة قاعدة المعطيات المستخدمة. يستخدم مخطط قاعدة المعطيات الناتج في مرحلة التصميم الفيزيائي حيث يجري تحديد الميزات الفيزيائية لقاعدة المعطيات.

9- تحويل مخطط E-R إلى جداول

يمكن تمثيل قاعدة معطيات موافقة لمخطط كيان -ارتباط بمجموعة جداول. فيوجد لكل صف كيانات ولكل صف ارتباطات جدول وحيد له نفس الاسم . يتألف كل جدول من مجموعة من الأعمدة لكل منها اسم وحيد (بعدد الواصفات).

إن كلا من النموذجين كيان-ارتباط والنموذج العلاقتي لقاعدة المعطيات هما مفهومان مجردان، يمثلان المخطط المنطقي للمؤسسة. ولما كان النموذجين يستخدمان مفاهيم

التصميم نفسها ، فإننا نستطيع التحويل من التصميم كيان-ارتباط إلى التصميم العلاقتي لقاعدة المعطيات.

9-1- تمثيل صفوف الكيانات كجدول

يجري تمثيل صف الكيانات القوي بجدول أعمدته هي نفس الواصفات الواصفة لصف الكيانات.

مثال :

<i>customer-name</i>	<i>social-security</i>	<i>c-street</i>	<i>c-city</i>
Jones	321-12-3123	Main	Harrison
Smith	019-28-3746	North	Rye
Hayes	677-89-9011	Main	Harrison

The *customer* table

أما صف الكيانات الضعيف فيجري تمثيله بواسطة جدول أعمدته تحوي الواصفات الواصفة للصف والأعمدة المكونة للمفتاح الأولي لصف الكيانات القوي المرتبط به.

مثال :

<i>loan-number</i>	<i>payment-number</i>	<i>payment-date</i>	<i>payment-amount</i>
L-17	5	10 May 1996	50
L-23	11	17 May 1996	75
L-15	22	23 May 1996	300

The *payment* table

9-2- تمثيل صفوف الارتباطات كجدول

يجري تمثيل صف الارتباط بين صفي كيانات كجدول أعمدته مؤلفة من مجموعة الأعمدة المكونة للمفتاحين الأولين لصفّي الكيانات المعنيين بالارتباط، ومجموعة أعمدة الوصفات الواصفة للعلاقة.

مثال :

<i>social-security</i>	<i>account-number</i>	<i>access-date</i>
...

The depositor table

9-3- الجداول المكررة

إن الجدول الناتج من علاقة الارتباط بين صف كيانات ضعيف وصف كيانات قوي هو مكرر، لأن المعلومات التي يحويها موجودة في الجدول الناتج من صف الكيانات الضعيف.

9-4- تمثيل التعميم كجداول

يوجد طريقتان للانتقال من مخطط E-R يحوي علاقة تعميم إلى جداول :

الطريقة 1 : توليد جدول لصف كيانات المستوى الأعلى، وتوليد جدول لكل صف كيانات من المستوى الأدنى يحوي أعمدة لكل واصفة من واصفات هذا الصف، إضافة إلى عمود لكل واصفة من واصفات المفتاح الأولي لصف كيانات المستوى الأعلى.

مثال

table	table attributes
<i>account</i>	<i>account-number, balance, account-type</i>
<i>savings-account</i>	<i>account-number, interest-rate</i>
<i>checking-account</i>	<i>account-number, overdraft-amount</i>

الطريقة 2 : إذا كانت علاقة التعميم منفصلة وتامة (لا يوجد كيان عضو في أحد صفوف الكيانات الدنيا مباشرة دون أن ينتمي إلى صف كيانات العليا، وكل كيان من المستوى الأعلى ينتمي إلى أحد صفوف كيانات المستوى الأدنى) عندئذ لا يوجد جدول لتمثيل صف كيانات المستوى الأعلى ويستعاض عنه بجدول لكل صف كيانات من المستوى الأدنى حاويا أعمدة لكل واصفة من واصفات صف الكيانات المعني وأعمدة لكل واصفة من واصفات صف كيانات المستوى الأعلى.

مثال :

table	table attributes
<i>savings-account</i>	<i>account-number, balance, interest-rate</i>
<i>checking-account</i>	<i>account-number, balance, overdraft-amount</i>

تمثيل علاقة ارتباط التجميع :

customer

<i>customer-name</i>	<u><i>cust-social-security</i></u>	<i>customer-street</i>	<i>customer-city</i>
----------------------	------------------------------------	------------------------	----------------------

loan

<u><i>loan-number</i></u>	<i>amount</i>
---------------------------	---------------

borrower

<u><i>cust-social-security</i></u>	<u><i>loan-number</i></u>
------------------------------------	---------------------------

employee

<u><i>emp-social-security</i></u>	<i>employee-name</i>	<i>phone-number</i>
-----------------------------------	----------------------	---------------------

loan-officer

<u><i>emp-social-security</i></u>	<u><i>cust-social-security</i></u>	<u><i>loan-number</i></u>
-----------------------------------	------------------------------------	---------------------------

تمارين الفصل الثاني

- 1- اشرح الفرق بين المفاهيم التالية : مفتاح أولي، مفتاح رئيسي، مفتاح مرشح
 - 2- تتعامل مديرية التسجيل والامتحانات في كلية المعلوماتية مع معطيات حول الصفوف، والدوام، والمدرسين، وأوقات وأمكنة إعطاء الدروس. كما تحتفظ المديرية بالعلامة التي يحصل عليها الطالب في كل مادة وتقدير الطالب في كل سنة دراسية. المطلوب إعطاء المخطط كيان-ارتباط لهذه القاعدة.
 - 3- لدى شركة تأمين مجموعة من الزبائن، يمتلك كل منهم سيارة أو أكثر. وتهتم شركة التأمين بحوادث السير التي تصيب كل سيارة من السيارات التي جرى التأمين عليها. المطلوب إعطاء المخطط كيان-ارتباط لهذه القاعدة.
 - 4- لدى مشفى الأسد الجامعي عدد من المرضى، ويعمل فيه عدد من الأطباء. يتألف المشفى من عدد من الأقسام التخصصية، ويجري قبول المرضى كل في القسم المتخصص في حالة هذا المريض. وهناك أيضا أقسام مشتركة تقدم الخدمات لكافة الأقسام مثل مخبر التحاليل الطبية، وقسم التصوير الشعاعي، والصيدلية. يخضع المريض خلال إقامته في المشفى لعدد من الفحوصات، وقد تجرى له عملية جراحية أو أكثر. لكل مريض من المرضى المقيمين في قسم معين طبيب مسؤول عن متابعته. ويكون هذا الطبيب واحدا من الأطباء العاملين في القسم.
- المطلوب :

أ. إعطاء المخطط كيان-ارتباط لهذه القاعدة

ب. ماذا يحصل لدى انتقال المريض من قسم إلى آخر، وكيف يمكن تمثيل ذلك في قاعدة المعطيات ؟

ت. كيف يمكن استرجاع السجل الطبي للمريض إذا راجع المشفى بعد مدة من خروجه من المشفى ؟

5- لدينا قاعدة معطيات خاصة ببرنامج الامتحان الأخير للجامعة، يمكن نمذجة هذه القاعدة كصف كيانات وحيد exam مع الوصفات course-name, section-number, room-number, time. كما يمكن بالمقابل أن نعرف مجموعة من صفوف كيانات مرتبطة بروابط للاستعاضة عن بعض الوصفات في صف الكيانات exam بالشكل :

course مع الوصفات name, departement, c-number.

Section مع الوصفات s-number, enrollment ومرتبطة كصف كيانات ضعيف مع course.

Room مع الوصفات r-number, capacity, building

والمطلوب :

أ. إعطاء مخطط E-R الذي يبين استخدام صفوف الكيانات الثلاثة المذكورة أعلاه.

ب. شرح كيف تحدد خواص التطبيق القرار في استخدام أحد صفوف الكيانات الإضافية في مخطط القاعدة.

6- المطلوب إعطاء تصميم لهرمية تعميم/تخصيص خاصة بشركة تباع السيارات والدرجات النارية. تباع الشركة الدراجات النارية، وسيارات النقل، والباصات، والشاحنات. مع تبرير وضع الوصفات كل مستوى من الهرمية. وشرح عدم إمكانية نقلها من مستوى إلى مستوى آخر.

7- المطلوب إنشاء المخطط التصميمي لشركة توزيع أدوية. تشتري هذه الشركة الأدوية من مصانع الأدوية وتخزنها في مستودعات موزعة في مراكز التوزيع الأساسية (في المحافظات)، ومن ثم توزعها إلى المتعاملين (صيادلة، مشافي). لكل دواء تتعامل به الشركة رقم مميز واسم تجاري واسم علمي (التركيب الكيميائي) وتاريخ انتهاء الصلاحية. تشتري الشركة الأدوية على شكل دفعات. تضم الدفعة عدة أدوية موزعة من معمل معين وتنتهي صلاحية الأدوية التي لها نفس الاسم التجاري في الدفعة الواحدة بنفس التاريخ (Lots). تصدر الشركة طلبات شراء للمعامل لتوريد الأدوية وتسدد قيم الأدوية بموجب فواتير (لكل فاتورة رقم ومجموعة أقلام وقيمة إجمالية). كما تتلقى الشركة طلبات شراء من المتعاملين وتقبض ثمن الأدوية المباعة بموجب فواتير.

8- المطلوب إنشاء المخطط التصميمي لشركة تصنيع أجهزة إلكترونية. تقوم الشركة بتصنيع الأجهزة على مرحلتين :

- تصنيع مكونات الأجهزة : البطاقات الإلكترونية، جسم الجهاز، ...
(منتجات شبه جاهزة). تصنع هذه المكونات من مواد أولية تشتريها من موردين.

• تجميع المكونات

تشتري هذه الشركة المواد الأولية من الموردين بموجب عقود. يتم توقيع العقد مع المورد لتوريد مجموعة من المواد الأولية، ويكلف أحد العاملين في قسم العقود بمتابعة العقد. يمكن توقيع أكثر من عقد مع مورد واحد. وتبيع الشركة منتجاتها إلى موزعين يشترون المنتجات الجاهزة فقط.

الفصل الثالث

النموذج العلاقاى

مقدمة

لقد فرض النموذج العلاقاى نفسه كأول نموذج للمعطيات لتطبيقات معالجة المعطيات الواسعة الانتشار. ففي قواعد المعطيات الأولى جرى اعتماد النموذج الهرمى أو الشبكى، لكن هذين النموذجين كانا شديدي الارتباط بطريقة التحقيق الفعلى للنظام، وهذا ما جعلهما أصعب استخداماً من النموذج العلاقاى.

وبسبب لوجود نظرية رياضية متكاملة تدعم النموذج العلاقاى، فإن ذلك يسهل تصميم قواعد المعطيات العلاقاتية وىتيح المعالجة الفعالة لطلبات المستخدمين.

1- بنية قواعد المعطيات العلاقاتية

تتألف قاعدة المعطيات من مجموعة من الجداول لكل منها اسم وحيد مميى. يتألف كل جدول بدوره من مجموعة من الأعمدة وعدد من الأسطر. يمثى كل سطر من الجدول علاقة تربط مجموعة من القيم. لما كان الجدول يتألف من مجموعة من هذه الارتباطات، فهناك تشابه شديد بين مفهوم الجدول ومفهوم العلاقة الرياضية، ومن هنا أخذ النموذج العلاقاى اسمه.

1-1- البنى الأساسية

انظر الشكل 1 الذي يبين جدول الحسابات account. يحوي هذا الجدول ثلاثة أعمدة هي :

- اسم الفرع branch_name
- رقم الحساب account_number
- الرصيد balance

وفق مصطلحات النموذج العلاقتي تسمى عناوين الأعمدة واصفات attributes. لكل واصف مجموعة من القيم تسمى مجال Domain الواصف. فمثلاً مجال الواصف "اسم الفرع" هو مجموعة كل أسماء الفروع، ولنرمز إلى هذه المجموعة بـ D1. ولنرمز بـ D2 إلى مجال الواصف account_number و بـ D3 إلى مجال الواصف balance.

إن كل سطر من الجدول account يتألف من ثلاثية (v1, v2, v3) حيث v1 هو اسم الفرع وينتمي إلى المجال D1، و v2 هو رقم الحساب وينتمي إلى المجال D2، و v3 هو الرصيد وينتمي إلى المجال D3.

branch name	account number	balance
Downtown	A-101	500
Mianus	A-215	700

الشكل 1 : العلاقة account

في الحالة العامة سوف يحوي الجدول account مجموعة جزئية من الجداء

$$D1 \times D2 \times D3$$

وعموماً فإن جدولاً يحوي n واصفاً يجب أن يكون مجموعة جزئية من الجداء

$$D1 \times D2 \times \dots \times Dn$$

يعرف الرياضيون العلاقة بأنها مجموعة جزئية من الجداء الديكارتي لمجموعة من المجالات. يتوافق هذا التعريف غالباً مع تعريفنا للجدول. الفرق هو فقط في أننا نعطي أسماء للواصفات، على حين يربط الرياضيون أرقاماً بالأسماء. فيستخدمون الرقم 1 للواصف الذي يظهر مجاله أولاً، والرقم 2 للواصف الذي يظهر مجاله ثانياً... وهكذا. وكما أن الجداول تُعبر عن العلاقات فإننا سوف نستخدم المصطلحات الرياضية (علاقة، حدودية) بدلاً من المصطلحات (جدول، سطر).

يوجد في العلاقة "account" التي تظهر في الشكل (1) سبع حدوديات، ولنفترض أن المتحول t من نوع حدودية يشير إلى أول حدودية في العلاقة. نستخدم المصطلح $t[\text{branch-name}]$ للدلالة على قيمة المتحول t في الواصف branch-name . وهكذا فإن $t[1] = \text{"Downtown"}$ and $t[\text{balance}] = 500$ وبالمقابل يمكننا كتابة $t[1]$ للدلالة على قيمة الحدودية t عند أول واصف branch_name . وكما أن العلاقة هي مجموعة من الحدوديات، فإننا سنستخدم المصطلح الرياضي $t \in r$ لنقول إن الحدودية t هي ضمن العلاقة r .

ملاحظات :

♦ في جميع العلاقات سنعتبر أن مجالات الواصفات المتعلقة بالعلاقة هي وحدوية atomic. نقول عن مجال إنه وحدوي إذا كانت عناصره وحدات غير قابلة للتجزئة.

مثال :

– مجموعة الأعداد هي مجال وحدوي (إلا في حالة اعتبار العدد سلسلةً من الأرقام)

– مجموعة مجموعات الأعداد هي مجال غير وحدوي.

♦ بإمكان عدة واصفات أن تأخذ قيمها من نفس المجال. مثال : مجموعة أسماء الأشخاص التي هي مجال للواصفين customer_name و employee_name .

- ♦ يوجد قيمة مشتركة في جميع المجالات وهي قيمة null التي تدل على أن القيمة غير معروفة أو ليست موجودة للواصف.

1-2- مخطط قاعدة المعطيات

يجب علينا التمييز بين مخطط قاعدة المعطيات database schema أو التصميم المنطقي لقاعدة المعطيات و حالة قاعدة المعطيات database instance التي هي صورة لمعطيات قاعدة المعطيات في لحظة محددة.

يشابه مفهوم العلاقة مفهوم التحولات في لغات البرمجة، على حين يشابه مفهوم مخطط العلاقة مفهوم تعريف الأنماط في لغات البرمجة.

ملاحظات :

- ♦ الاسم المعطى لمخطط العلاقة يبدأ بحرف كبير والاسم المعطى للعلاقة هو بأحرف صغيرة.

مثال : نستخدم Account-schema للدلالة على مخطط العلاقة للعلاقة account ونكتب :

Account-schema = (branch-name, account-number, balance)

وندل على أن account هي علاقة على Account-schema بـ

account(Account-schema)

- ♦ وبوجه عام نقول إن مخطط العلاقة يحوي قائمة من الواصفات ومجالاتها المعتبرة.

- ♦ يُعبر مفهوم حالة علاقة relation instance عن مفهوم قيمة التحول في لغات البرمجة. وغالباً ما نستخدم كلمة "علاقة" ونعني بها "حالة العلاقة" relation instance.

1-3- المفاتيح

إن مفاهيم المفتاح الرئيسي، والمفتاح المرشح والمفتاح الأولي التي ناقشناها سابقاً مطبقة في النموذج العلاقتي.

لتكن R مخطط علاقة. إذا قلنا إن K ، وهي مجموعة جزئية من R ، هي مفتاح رئيسي لـ R نكون قد اعتبرنا أنه في جميع العلاقات $r(R)$ لا توجد حدوديتان لهما نفس القيم لجميع الواصفات الموجودة في K . أي إذا كان t_1 و t_2 من r و $t_1 \supset t_2$ فإن $t_1[K] \supset t_2[K]$.

إذا كان مخطط قاعدة المعطيات العلاقتي معتمداً على الجداول المشتقة من مخطط كيان-ارتباط فإنه بالإمكان تحديد المفتاح الأولي للمخطط العلاقتي من المفاتيح الأولية للكيانات وصف الارتباطات التي اشتق منها هذا المخطط كما يلي :

صف الكيانات القوي : المفتاح الأولي للصف هو المفتاح الأولي للعلاقة.

صف الكيانات الضعيف : المفتاح الأولي للعلاقة الناتجة يتألف من اجتماع المفتاح الأولي لصف الكيانات القوي المرتبط به ، والمميز لصف الكيانات الضعيف.

صف الارتباطات : إذا كانت العلاقة كثير-كثير فإن المفتاح الأولي لها هو اجتماع المفتاحين الأولين لصفي الكيانات المرتبطة بهذه العلاقة. وإذا كانت العلاقة من نوع كثير-واحد من $A \rightarrow B$ فإن المفتاح الأولي للعلاقة يتكون من المفتاح الأولي لصف الكيانات الكثير A .

1-4- لغات الاستعلام

لغة الاستعلام هي اللغة التي يطلب بواسطتها المستخدم معلومات من قاعدة المعطيات. وهي عادة من مستوى أعلى من لغات البرمجة القياسية.

يمكن تصنيف هذه اللغات في نوعين : لغات إجرائية ولغات غير إجرائية.

اللغات الإجرائية : يقوم فيها المستثمر بتحديد مجموعة من العمليات التي يجريها النظام على قاعدة المعطيات للوصول إلى المعطيات المرغوبة.

اللغات غير الإجرائية : يصف فيها المستثمر المعلومات المرغوبة دون إعطاء الإجراء المحدد للحصول عليها.

تقدم معظم نظم قواعد المعطيات التجارية كلتا اللغتين. سنقوم بدراسة بعض هذه اللغات.

2- الجبر العلاقتي

الجبر العلاقتي هو لغة استعمال إجرائية. يتألف من مجموعة من العمليات التي تأخذ علاقة أو اثنتين كدخل، وتعطي علاقة جديدة كخرج. العمليات الأساسية في الجبر العلاقتي هي :

- الاختيار Select
- الإسقاط Project
- الاجتماع Union
- الفرق Difference
- الجداء الديكارتي Cartesian product
- إعادة التسمية Rename
- إضافة إلى العمليات المعرفة ابتداءً من العمليات الأساسية :
- التقاطع Intersection
- الدمج الطبيعي Natural Join
- القسمة Division
- النسب Assignment

سوف نعرض فيما يلي العمليات الأساسية والإضافية في الجبر العلاقائي موضحين طريقة عملها بأمثلة على قاعدة المعطيات المتعلقة بالمصارف والمعرفة بالمخطط العلاقائي التالي :

Loan-schema (loan-number, amount, branch-name)

Customer-schema(customer-name, customer-street, customer-city)

Borrower-schema(customer-name, loan-number)

Employee-schema(employee-name, phone-number)

Loan-officer-schema(banker-name, customer-name, loan-number)

Depositor-schema (customer-name, account-number)

Account-schema(account-number, balance)

Branch-schema(branch-name, branch-city, assets)

2-1- العمليات الأساسية

تسمى العمليات (اختيار، إسقاط، إعادة التسمية) عمليات أحادية لأنها تجرى على علاقة واحدة. العمليات الثلاث الباقية هي ثنائية لأنها تجرى على زوج من العلاقات.

♦ عملية الاختيار

تقوم باختيار مجموعة من الحدوديات التي تحقق شرطا معيناً من علاقة. سنرمز إلى العملية كما يلي :

$$\sigma_{condition}(relation)$$

حيث condition هو شرط يجب أن تحققه الحدوديات المختارة.

مثال : لناخذ مخطط العلاقة التالية

Loan-schema (branch-name, loan-number, amount)

لاختيار مجموعة الحدوديات القروض للفرع "perryridge" branch-name = نكتب
العبارة التالية :

$$\sigma_{branch-name="perryridge"}(loan)$$

يمكن أن يحوي الشرط المطبق في عملية الاختيار عمليات مقارنة : = ، < ، > ،
= ، < ، > ومعاملات منطقية and, or, not ويمكن أن تطبق هذه المعاملات بين قيم
الواصفات المكونة للعلاقة.

مثال : لاختيار القروض التي منحها الفرع "perryridge" branch-name = والتي لا
تقل عن 1200 نكتب :

$$\sigma_{branch-name="perryridge" \wedge amount \geq 1200}(loan)$$

لاختيار مجموعة الزبائن الذين يحملون نفس اسم المسؤول عن القرض الذي اقترضوه من
علاقة loan-officer ذات المخطط التالي

$$Loan-officer = (customer-name, banker-name, loan-number)$$

نكتب :

$$\sigma_{customer-name=banker-name}(loan-officer)$$

♦ عملية الإسقاط projection

وهي عملية وحيدة المعامل تسمح بانتقاء بعض الوصفات من العلاقة. نرمز إلى هذه العملية بالشكل :

$$\Pi_{selected \ attributes}(relation)$$

مثال : لنفترض أننا نريد الحصول على أرقام القروض ومبالغها دون أن نهتم بالأسماء الأفرع.

يكتب الاستعلام السابق بالشكل :

$$\Pi_{loan-number, amount}(loan)$$

وبفرض أن علاقة القروض loan ممثلة بالجدول التالي :

loan-number	branch-name	amount

تكون العلاقة الناتجة من عملية الإسقاط من الشكل :

loan-number	amount

♦ تركيب العمليات العلاقتية

إن نتيجة العمليات العلاقتية هي علاقة. هذا ما يسمح لهذه العمليات أن تجتمع لتؤلف عبارات الجبر العلاقتي.

مثال : لإيجاد أسماء الزبائن الذين يعيشون في مدينة "Harrison".

نكتب :

$$\Pi_{customer-name} (\sigma_{customer-city="Harrison"}(customer))$$

ونقصد بها تركيب عمليتين : اختيار الحدوديات التي تحقق الشرط (مدينة = "Harrison") وإسقاط العلاقة الناتجة من العملية السابقة على العمود customer-name .

♦ عملية الاجتماع Union Operation

نرمز إلى العملية بـ \cup وتُجرى هذه العملية بين علاقيتين متجانستين فنقول إنه يمكننا القيام بالعملية $r \cup s$ إذا تحقق الشرطان التاليان :

- عدد الواصفات في العلاقتين r و s هو نفسه ،
- مجال الواصفة رقم i في r هو نفسه مجال الواصفة رقم i في s .

ويمكن بالطبع أن تكون العلاقتان r, s ناتجتين عن تعبير في جبر علاقتي.
مثال : للحصول على جميع الزبائن الذين يتعاملون مع المصرف (الذين لديهم حساب أو اقترضوا قرضاً أو للحصول على كلا الفريقين).
نحتاج إلى إيجاد مجموعة الزبائن الذين لديهم حساب في المصرف ، وهي معلومات موجودة في علاقة depositor ، وإلى إيجاد مجموعة الزبائن الذين اقترضوا من المصرف ، وهي معلومات موجودة في علاقة borrower ثم إلى إجراء عملية الاجتماع بين المجموعتين.

ومن ثم يكون التعبير الناتج هو :

$$\Pi_{customer-name} (depositor) \cup \Pi_{customer-name} (borrower)$$

♦ عملية الفرق Difference Operation

نرمز إلى هذه العملية بـ " - " وتسمح بإيجاد الحدوديات التي تنتمي إلى علاقة الحد الأول ولا تنتمي إلى علاقة الحد الثاني.
 لإيجاد مجموعة الزبائن الذين لديهم حساب مصرفي ولم يقرضوا من المصرف نكتب :

$$\prod_{\text{customer-name}}(\text{depositor}) - \prod_{\text{customer-name}}(\text{borrower})$$

وكما ذكرنا في عملية الاجتماع لكي تجري عملية الفرق بين علاقيتين يجب أن يتحقق الشرطان المذكوران في الفقرة السابقة.

♦ عملية الجداء الديكارتي Cartesian Product

نرمز إلى هذه العملية بـ X وتسمح بتجميع معلومات من علاقيتين ونكتب $r1 \times r2$ وبوجه عام نقول :

إذا كان لدينا العلاقتان $r1 (R1)$, $r2 (R2)$ فإن $r1 \times r2$ هي علاقة R مخططها العلائقي هو تلاصق $R1$ و $R2$ وتحوي جميع الحدوديات t التي تحقق الشرط التالي :

يوجد $t1$ من $r1$ و $t2$ من $r2$ بحيث

$$t[R1] = t1[R1] \wedge t[R2] = t2[R2]$$

مثال :

– ليكن لدينا R1 و R2 بحيث $R = R1 \times R2$

R	A	B	C	D
	a1	b1	c1	d1
	a1	b1	c2	d2
	a2	b2	c1	d1
	a2	b2	c2	d2

R1	A	B
	a1	b1
	a2	b2

R2	C	D
	c1	d1
	c2	d2

– إذا أردنا الحصول على جميع الزبائن الذين اقترضوا من المصرف الفرع "perryridge".

للحصول على هذه المعلومات نحتاج إلى المعلومات الموجودة في كلتا العلاقتين loan و borrower وتعطي عملية اختيار الحدوديات، المتعلقة بالفرع المطلوب من جداء العلاقتين، معلومات عن الزبائن والقروض المأخوذة من الفرع المطلوب، ولكن ليست المعلومات المطلوبة، وللحصول على المعلومات المطلوبة نكتب :

$$\Pi_{\text{customer-name}}(\sigma_{\text{borrower-loan-number=loan-number}}(\sigma_{\text{branch-name="perryridge"}}(\text{borrower} \times \text{loan})))$$

♦ إعادة التسمية Rename

من المفيد إعطاء أسماء للعلاقات الناتجة عن تعبير جبر علاقتي. نرمز إلى العملية بالرمز :

$$\rho_x(E)$$

التي تعني أن نتيجة التعبير E توضع في العلاقة x.

لنبيين استخدام هذه العملية بالمثل التالي : لإيجاد الرصيد الأعلى في المصرف،
نستخدم الاستراتيجية التالية :

- إيجاد مجموعة الأرصدة غير العظمى للحسابات في المصرف،
- طرح المجموعة الناتجة من مجموعة الأرصدة في المصرف فنحصل علي الرصيد
الأعظم المطلوب .

مجموعة الأرصدة الموجودة في المصرف هي :

$$\Pi_{balance}(account)$$

مجموعة الأرصدة غير العظمى هي :

$$\Pi_{account-balance}(\sigma_{account-balance < d\ balance}(account \times \rho_d(account)))$$

والنتيجة هي :

$$\Pi_{balance}(account) - \Pi_{account-balance}(\sigma_{account-balance < d\ balance}(account \times \rho_d(account)))$$

2-2- العمليات الإضافية

تسمح العمليات التي رأيناها حتى الآن بإعطاء تعريف كامل لأي تعبير في الجبر العلاقاقي. ولكن إذا قصرنا أنفسنا على هذه العمليات فالتعبير عن بعض الاستعلامات سيكون طويلاً نسبياً. ولذلك جرى تعريف بعض العمليات الإضافية التي لا تؤثر في قوة الجبر العلاقاقي، ولكنها تبسط التعابير المثلثة للاستعلامات.

♦ عملية التقاطع

تُجرى هذه العملية بين علاقيتين ويجب أن تحقق هاتان العلاقتان الشروط المذكورة في عملية الاجتماع. ونتيجة عملية التقاطع هي علاقة تحوي مجموعة الحدوديات الموجودة ضمن العلاقتين. التعبير الموافق لهذه العملية هو :

$$R \cap S = R - (R - S)$$

♦ عملية الدمج الطبيعي Natural join operation

غالباً ما نرغب في تبسيط بعض الاستعلامات التي تحتاج إلى جداء ديكارتي، ومعظم عمليات الاستعلام التي تحوي جداءاً ديكارتيّاً تحوي عملية اختيار من نتيجة الجداء. فعملية الدمج هي عملية ثنائية تسمح بتركيب عملية الاختيار والجداء الديكارتي بعملية واحدة.

لنأخذ كتعريف لهذه العملية العلاقتين $r(R)$ و $s(S)$ ونقول إن دمج العلاقتين هي علاقة مخططها هو اجتماع مخططي العلاقتين ومعرفة بالشكل :

$$r \bowtie s = \prod_{RYS} (\sigma_{rA1=sA1 \wedge rA2=sA2 \wedge \dots \wedge rAn=sAn} (r \times s))$$

حيث

$$R \cap S = \{A1, A2, \dots, An\}$$

مثال :

لإيجاد أسماء جميع الأفرع التي لزيائنها حساب في المصرف وتعيش في مدينة "Harrison" نكتب.

$$\prod_{branch-name} (\sigma_{customer-city="Harrison"} (customer \bowtie account \bowtie depositor))$$

حالات خاصة :

- إذا كان

$$R \mid S = \Phi \Rightarrow r \times s = r \times s$$

وكانت θ هناك قضية على الواصفات في مخطط العلاقة الناتجة عن الدمج نكتب :

$$r \times_{\theta} s = \sigma_{\theta}(r \times s)$$

♦ عملية القسمة

هذه العملية مناسبة للاستعلامات التي تحوي كلمة "لأجل كل". فلإيجاد مجموعة الزبائن الذين لهم حسابات مصرفية في جميع الأفرع الموجودة في مدينة "brooklyn" نقوم بما يلي :

نستخرج مجموعة الفروع الموجودة في مدينة "brooklyn" بكتابة التعبير التالي :

$$r1 = \prod_{branch-name} (\sigma_{branch-city="brooklyn"}(branch))$$

ثم نستخرج مجموعة الزبائن والفروع الذين لديهم حسابات فيها بكتابة التعبير :

$$r2 = \prod_{customer-name, branch-name} (depositor \times account)$$

ونحصل على النتيجة المطلوبة بكتابة :

$$r2 \div r1$$

وكتعريف لعملية القسمة :

ليكن لدينا $r(R)$ و $s(S)$ علاقتان بحيث إن المخطط العلاقتي $\downarrow S$ هو جزء من المخطط العلاقتي $\downarrow R$. فالعلاقة الناتجة عن خارج قسمة r على s هي علاقة لها المخطط العلاقتي $R-S$ وحدودياتها t تحقق الشرطين التاليين :

- t هي من العلاقة

$$\Pi_{R-S}(r)$$

- في حالة جميع حدوديات ts من s يوجد حدودية tr من r بحيث :

$$tr[S] = ts[S]$$

$$tr[R-S] = t$$

♦ عملية الإسناد

تعمل هذه العملية بكيفية مشابهة لعملية الإسناد في لغات البرمجة، وتعتبر طريقة مناسبة للتعبير عن استعلامات معقدة.

يرمز إلى العملية بـ " \leftarrow "

3- لغة القضايا The tuple Relational Calculus

هي لغة استعلام غير إجرائية تسمح بوصف المعلومات المرغوب الحصول عليها دون تبيان الإجراء اللازم لذلك. الاستعلام بهذه اللغة له الشكل التالي $\{t \mid p(t)\}$ الذي يعني : المطلوب الحصول على جميع الحدوديات t التي تحقق القضية p .

يستخدم في هذه اللغة الرموز التالية :

$t[A]$ للدلالة على قيمة الحدودية t في الوصفة A .

$t \in r$ للدلالة على أن الحدودية t هي في العلاقة r . t متحول حدودي

\exists يوجد

\forall مهما يكن

المتحول الحدودي هو متحول حر ماعدا المتحول المسبوق بمعامل "يوجد" أو "مهما يكن".

والقضية P مبنية على تركيب من عناصر لها أحد الأشكال التالية :

- $s \in r$ حيث s متحول حدودي و r علاقة.
- $s[x] \circ u[y]$ حيث s, u متحويلات حدودية ، x واصفة في s و y واصفة في u و \circ عملية مقارنة ويجب أن يكون مجالا الواصفتين x, y متساويين.
- $s[x] \circ c$ حيث c ثابت في مجال تعريف الوصفة x .

ويجري بناء القضايا باستخدام القواعد التالية :

- أي عنصر معرف بأحد الأشكال السابقة هو قضية .
- p قضية $\Leftarrow (p)$ و $\neg p$ هما قضيتان
- $P1, P2$ قضيتان $\Leftarrow P1 \Rightarrow P2, P1 \vee P2, P1 \wedge P2$ جميعها قضايا
- إذا كان $P1(s)$ قضية حيث s متحول حر و r علاقة فإن $\exists s \in r(P1(s))$ و $\forall s \in r(P1(s))$ قضايا

أمثلة :

- المطلوب إيجاد أسماء الفروع مع القروض الموجودة فيها والتي تزيد عن 1200 ؟

الصياغة بلغة القضايا هي بالشكل :

$$\{t \mid t \in \text{loan} \wedge t[\text{amount}] > 1200\}$$

- وإيجاد جميع الزبائن المتعاملين مع المصرف (قرض أو حساب) نكتب :

$$\{t \mid \exists s \in \text{borrower}(t[\text{customer-name}] = s[\text{customer-name}]) \\ \vee \exists u \in \text{depositor}(t[\text{customer-name}] = s[\text{customer-name}])\}$$

4- العمليات الموسعة للجبر العلاقتي

جرى توسيع عمليات الجبر العلاقتي بعدة طرق. فأجري توسيع بسيط يجعل العمليات الحسابية جزء من عملية الإسقاط، وتوسيع بالسماح بالقيام بعمليات التجميع مثل : حساب مجموع عناصر مجموعة Sum أو المتوسط الحسابي لعناصر مجموعة Average، وتوسيع بإضافة عملية الدمج الخارجي outer join التي تسمح بالتعامل مع قيم غير المحددة null والتي تُنمذج المعلومات الناقصة.

4-1- الإسقاط المعمم

عملية الإسقاط المعمم هي عملية الإسقاط مع استخدام التوابع الرياضية في قائمة الإسقاط. يُعبر عن هذه العملية بالشكل :

$$\prod_{F1, F2, F3, \dots, Fn} (E)$$

حيث : E تعبير بالجبر العلاقتي،

F1, F2, ..., Fn تعابير حسابية بين واصفات وثوابت في المخطط العلاقتي لـ E.

وكحالة خاصة يمكن أن يكون التعبير الرياضي شكلاً بسيطاً واصفة أو ثابت.

مثال :

لنأخذ العلاقة credit-info التي مخططها العلاقتي هو :

Credit-info-schema = (customer-name, limit, credit-balance)

والمطلوب حساب الكمية المتبقية لكل زبون ؟

نكتب التعبير التالي :

$$\prod_{customer-name, limit-credit-balance} (credit-info)$$

4-2- الدمج الخارجي

لنأخذ العلاقات ذات المخططات العلاقتية التالية :

Employee (employee-name, street, city)

Ft-works (employee-name, branch-name, salary)

ولنفترض أننا نريد توليد علاقة واحدة تحوي جميع المعلومات حول الموظفين ؟

بالإمكان استخدام عملية الدمج الطبيعي :

employee	employee-name	street	city	ft-works	employee-name	branch-name	salary
	A	S1	C1		A	Br1	500
	B	S2	C1		B	Br2	700
	C	S4	C2		D	Br1	1220
	D	S5	C3		F	Br3	2000

تعطي عملية الدمج الطبيعي النتيجة التالية :

employee	ft-works	employee-name	street	city	branch-name	salary
		A	S1	C1	Br1	500
		B	S2	C1	Br2	700
		D	S5	C3	Br1	1220

إن لا توجد معلومات حول الموظف C في العلاقة ft-works، ولا توجد معلومات حول الموظف F في العلاقة employee. نستخدم عملية الدمج الخارجي لتجنب ضياع المعلومات الموجودة، ويوجد ثلاثة أنواع من هذه العملية:

- الدمج الخارجي اليساري ويرمز إليها بـ \bowtie ،
- الدمج الخارجي اليميني ويرمز إليها بـ \ltimes ،
- الدمج الخارجي الكامل ويرمز إليها بـ \ltimes .

تمثل الأشكال الثلاثة عملية الدمج مع إضافة حدوديات إلى نتيجة عملية الدمج الطبيعي. ففي عملية الدمج اليساري left outer join تتصلب إلى نتيجة الدمج الطبيعي حدوديات موجودة في العلاقة اليسارية وغير موجودة في العلاقة اليمينية، بعد وضع قيمة null عدم التعيين للواصفات القادمة من العلاقة اليمينية. تُجرى عملية الدمج الخارجي اليميني right outer join بكيفية مشابهة لعملية الدمج الخارجي اليساري، فتؤخذ الحدوديات الموجودة في العلاقة اليمينية وغير موجودة في العلاقة اليسارية. أما عملية الدمج الخارجي الكامل فهي اجتماع عمليتي الدمج الخارجي اليميني واليساري.

تمثل الجداول التالية العمليات السابقة :

employee \bowtie ft-works

employee-name	street	city	branch-name	salary
A	S1	C1	Br1	500
B	S2	C1	Br2	700
D	S5	C3	Br1	1220
C	S4	C2	Null	Null

الدمج الخارجي اليساري

Employee \bowtie ft-works

employ ee-name	street	city	branch- name	salary
A	S1	C1	Br1	500
B	S2	C1	Br2	700
D	S5	C3	Br1	1220
F	Null	Null	Br3	2000

عملية الدمج الخارجي اليميني

Employee \bowtie ft-works

employee -name	street	city	branch- name	salary
A	S1	C1	Br1	500
B	S2	C1	Br2	700
D	S5	C3	Br1	1220
F	Null	Null	Br3	2000
C	S4	C2	Null	Null

عملية الدمج الخارجي الكامل

4-3- التتابعات التجميعية Aggregate functions

تسمح التتابعات التجميعية بأخذ مجموعة من القيم وإعادتها كقيمة وحيدة. من هذه التتابعات :

- تابع الجمع Sum
- تابع المتوسط الحسابي Avg
- تابع تعداد عدد العناصر Count
- تابع أصغر قيمة Min
- تابع أكبر قيمة Max
- عملية تجميع العناصر في مجموعات جزئية group

مثال : إيجاد مجموع رواتب العاملين ft-works.

نكتب `Sum salary (ft-works)`

ولحذف التكرار في القيم الموجودة أو المضمنة تضاف كلمة `distinct` إلى العملية ، فمثلا

لإيجاد عدد الأفرع التي يتعامل معها الموظفون بدوام كامل نكتب :

`Count-distinct branch-name (ft-works)`

ولإجراء عملية جمع لرواتب الموظفين بدوام كامل والمجموعين ضمن الفروع نكتب :

`branch-name G Sum salary (ft-works)`

حيث الشكل العام لعملية التجميع هو : $G_1, G_2, G_3, \dots, G_n \text{ FIA}_1, \text{FIA}_2, \dots, \text{FIA}_m (E)$

E تعبير جبر علاقتي ،

$G_1, G_2, G_3, \dots, G_n$ تؤلف قائمة من الواصفات التي سيجري التجميع وفقها ،

F_i هو تابع تجميعي وكل A_i هي اسم واصفة.

ونتيجة العملية هي أن مجموعة الحدوديات الناتجة من التعبير E تقسم إلى مجموعات بحيث تكون مجموعة الحدوديات المنتمية إلى مجموعة تملك قيما متساوية لقائمة الواصفات $G1, G2, G3, \dots, Gn$.

5- تعديل قاعدة المعطيات

بعد دراسة كيفية استخراج المعلومات من قاعدة المعطيات، سنورد كيفية التعامل مع هذه المعلومات من إضافة وحذف وتعديل. سنقوم بهذه العمليات باستخدام عملية الإسناد.

5-1 الحذف Deletion

يعبر عن طلب الحذف بنفس طريقة الاستعلام، وبدلاً من ظهور الحدوديات للمستثمر سوف تحذف الحدوديات الناتجة من قاعدة المعطيات. نعبر عن عملية الحذف في الجبر العلائقي بـ $r \leftarrow r - E$ حيث r علاقة، E تعبير استعلام في الجبر العلائقي.

مثال :

- لحذف جميع الحسابات المصرفية الخاصة بالسيد "Smith" نكتب :

$$account \leftarrow account - \sigma_{customer-name='Smith'}(account)$$

- لحذف جميع الحسابات الموجودة في الأفرع الموجودة في مدينة "Needham".

$$r1 \leftarrow \sigma_{branch-city='Needham'}(account \times branch)$$

$$r2 \leftarrow \Pi_{branch-name, account_number, balance}(r1)$$

حيث استخدمت علاقات وسيطة.

$$account \leftarrow account - r2$$

5-2- الإضافة Insertion

تجري عملية الإضافة إلى علاقة عن طريق تحديد الحدوديات المراد إضافتها، بتحديد قيم الواصفات لكل حدودية (بحيث تكون هذه القيم ضمن مجال تعريف الواصفات) أو بإعطاء الاستعلام الذي نتيجته مجموعة حدوديات متوافقة مع العلاقة (من حيث عدد الواصفات ومجال تعريف كل منها). الشكل العام لعملية الإضافة في الجبر العلاقتي هو:

$$r \leftarrow r \cup E$$

حيث r علاقة و E تعبير جبر علاقتي.

أمثلة :

لإضافة حساب في علاقة $account$ إلى الشخص $Smith$ نكتب :

$$account \leftarrow account \cup \{("Perryridg", A-932,1200)\}$$

$$depositor \leftarrow depositor \cup \{("Smith", A-932)\}$$

5-3- التعديل Updates

لتعديل قيمة أو مجموعة قيم متعلقة بواصفات حدودية معينة نستخدم عملية الإسقاط المعمم كالتالي :

$$r \leftarrow \prod_{F1, F2, \dots, Fn} (r)$$

حيث F_i تعبير للواصفة رقم i إذا طرأ تعديل عليها، ويساوي A_i في حال عدم وجود أي تعديل.

مثال : لنفترض أن المطلوب تعديل أرصدة الحسابات الموجودة في المصرف بإضافة فائدة تقدر بـ 5٪ إلى الرصيد.

تجري عملية التعديل كما يلي :

$$account \leftarrow \prod_{branch-name, account-number, balance \leftarrow balance * 1.05} (account)$$

6- المنظار Views

حتى لحظتنا هذه نتعامل مع المستوى المنطقي لقاعدة المعطيات، الذي عبرنا عنه بتجميع من المخططات العلاقية المؤلفة للمعلومات المخزنة في قاعدة المعطيات العلاقية. إن التعامل المباشر لجميع المستثمرين مع المستوى المنطقي تعاملًا كاملاً غير مستحب لاعتبارات أمنية، فمن المرغوب فيه إمكان حجب بعض المعلومات عن بعض المستثمرين. أو توليد مجموعة علاقات خاصة بمجموعة مستثمرين. انطلاقاً من ذلك فقد أضيف إمكان تعريف منظار لجزء من المخطط المنطقي لقاعدة المعطيات، ويظهر هذا الجزء للمستخدم كعلاقة وهمية.

♦ تستخدم التعليمة التالية لتعريف المنظار :

create view v as <query expression>

حيث v اسم المنظار المراد تعريفه، و $query\ expression$ هو تعبير استعلام مبني على العلاقات المراد تعريف منظار لها.

مثال : لتعريف منظار لجميع المتعاملين مع المصرف (مقرضين أو لديهم حساب مصرفي) والفروع الذين يتعاملون معها.

نكتب :

create view all-customer as

$$\prod_{branch-name, customer-name} (depositor \times account) \cup \\ \prod_{branch-name, customer-name} (borrower \times account)$$

♦ بعد تعريف المنظار يمكن استخدام اسم المنظار للدلالة على علاقة وهمية ، ومن ثم يمكننا اختيار حدوديات منها.

مثال : لإيجاد أسماء المتعاملين مع المصرف في الفرع perryridge نكتب :

$$\prod_{customer-name} (\prod_{branch-name="perryridge"} (all - customer))$$

الفرق الأساسي بين عملية الإسناد وعملية تعريف المنظار، هو أن العلاقة الناتجة من عملية الإسناد، تحوي معلومات لا تتغير بتغير المعلومات الموجودة في العلاقات التي تدخل في التعبير المكون لعملية الإسناد. على حين نجد أن محتوى العلاقة الوهمية المكونة للمنظار يتبدل بتبدل محتوى العلاقات المشكلة للمنظار.

وبوجه عام تقوم معظم نظم إدارة قواعد المعطيات بتخزين التعبير المعروف بالمنظار وفقه وليس نتيجة التعبير الجبري، وبعضها يسمح بتخزين علاقة المنظار. وعند حصول أي تعديل على محتوى العلاقات الداخلة في تشكيل المنظار تجرى هذه التعديلات مباشرة على علاقة المنظار.

♦ يمكن تعرف منظار باستخدام مناظير معرفة سابقا. مثال :

create view perryridige-customer as

$$\prod_{customer-name} (\sigma_{branch-name} (all - customer))$$

♦ تظهر بعض المشاكل عند إجراء تعديلات من خلال المنظار، ذلك أن أي تعديل على المنظار سيترجم إلى مجموعة من التعديلات على العلاقات الأصلية المشكلة للمنظار. لنبين ذلك بالمثال التالي :

لنفترض أننا عرفنا منظارا لرؤية المعلومات عن القروض في المصرف، عدا ما يتعلق بمعلومة كمية القرض كما يلي :

create view branch-loan as $\prod_{branch-name, loan-number} (loan)$

ولنفترض أننا نريد تعديل المنظار بإضافة قرض جديد وليكن ("perryridge", L-37) . إن هذه العملية ستترجم إلى عملية إضافة إلى العلاقة الأصلية loan ولكن ضمن الحدودية المراد إضافتها لا توجد لدينا معلومات عن كمية القرض، وبمن ثم يقوم النظام بتنفيذ أحد الحلين التاليين :

- إعادة رسالة خطأ وعدم إجراء العملية.

- إضافة الحدودية التالية إلى علاقة loan : ("perryridge", L-37, null) .

7- الخلاصة

رأينا أن النموذج العلاقتي قائم على تجمع من الجداول، يمكن للمستثمر الاستعلام من خلالها، وإضافة وتعديل وحذف حدوديات منها. يجري التعبير عن هذه العمليات بواسطة عدة لغات. درسنا منها لغة الجبر العلاقتي، وهي لغة إجرائية تعرف العمليات الأساسية المستخدمة في لغة الاستعلام العلاقتي، ولغة القضايا وهي لغة غير إجرائية.

المنظار هو علاقة وهمية تعرف بتعبير استعمال، وهو تقنية مفيدة لتبسيط بعض الاستعلامات، ولكن يمكن أن ينتج بعض المشاكل لدى إجراء تعديل قاعدة المعطيات من خلال المنظار. يجري تخزين المنظار فيزيائيا لزيادة فعالية الاستعلام من خلاله وعندها يجري تعديل محتوى المنظار بعمليات إضافية لدى حصول أي تعديل على قاعدة المعطيات.

تمارين الفصل الثالث

لتكن قاعدة المعطيات العلاقتية المعرفة بالمخططات العلاقتية التالية :

employee (employee-name, street, city)
works(employee-name, company-name,salary)
company(company-name, city)
manages(employee-name, manager-name)

والمطلوب إعطاء التعبير الموافق بلغة الجبر العلاقتي لكل مما يلي :

1. أسماء جميع الموظفين العاملين في المصرف التجاري.
2. أسماء ومدن إقامة جميع الموظفين العاملين في المصرف التجاري والذين يكسبون أكثر من 100000 ل. س في السنة
3. أسماء جميع الموظفين الذين يقطنون في نفس المدينة التي توجد فيها الشركة التي يعملون فيها.
4. أسماء جميع الموظفين الذين يقطنون في نفس المدينة والشارع الذين يقطن فيهما مدراؤهم.
5. أسماء الشركات الموجودة في عدة مدن.
6. أسماء الشركات الموجودة في مدن للمصرف التجاري تواجد فيها.
7. أسماء الموظفين الذين رواتبهم أعلى من رواتب جميع موظفي المصرف التجاري.
8. حذف جميع الحدوديات الموجودة في علاقة works والمتعلقة بموظفي المصرف التجاري.

الفصل الرابع

لغة SQL

مقدمة

درسنا في الفصل السابق لغة استعمال تقدم العمليات اللازمة لتمثيل استفسار معين. ولما كانت نظم إدارة قواعد المعطيات التجارية تتطلب لغة استفسار أكثر قرباً من المستثمر، فإننا سندرس في هذا الفصل أكثر لغات الاستعمال التجارية انتشاراً وهي لغة SQL ، والتي هي تركيب من لغة الجبر العلاقتي والحساب العلاقتي. تجمع لغة SQL إمكانات إضافية إلى الاستعمال من قاعدة المعطيات، فتسمح بتعريف بنية المعطيات وإضافة وتعديل المعطيات في قاعدة المعطيات، وتحديد أمانها. لن نقوم بتقديم دليل استخدام كامل لـ SQL ، وسنقوم فقط بعرض البنى والمفاهيم الأساسية لهذه اللغة.

لمحة تاريخية

طُوِّرَت النسخة الأصلية من SQL مخبر البحث IBM San Jose وأطلق عليها اسم Sequel وكانت جزءاً من مشروع نظام R في بداية 1970 ، ثم جرى تطوير هذه اللغة وتغيير اسمها إلى S QL (Structured Query Language).

في عام 1986 نشر معهد المقياس الأمريكي ANSI وهيئة المقياس العالمية (ISO) لغة SQL المعيارية. سُميت باسم SQL-86.

كما قامت IBM بنشر معيار SQL خاص بها وتوسّع لهذا المعيار SQL المعياري عام 89، ثمّ اعتمدت نظم قواعد المعطيات هذه اللغة حتى اليوم.

سنعرض فيما يلي لمحة إلى SQL المعتمدة على المعيارين SQL-89 و SQL-92.

1- تعريف بلغة SQL

تتكون لغة SQL من عدة أجزاء:

- لغة تعريف المعطيات (Data Definition Language) DDL : وتقدم التعليمات اللازمة لتعريف وتعديل مخطط علاقة، حذف علاقة، بناء فهرس .
- لغة التعامل مع المعطيات (Interactive Data Manipulation Language) (DML Language) : وتعتمد طريقةً لصياغة الاستعلامات المرتكزة على الجبر العلائقي وجبر القضايا، وتحوي تعليمات لإضافة، وحذف وتعديل حدوديات في قاعدة المعطيات.
- لغة التعامل مع المعطيات المضمنة Embedded : وهي مصممة للتضمين في لغات البرمجة الاعتيادية مثل PL/I، باسكال، C، كوبول.
- تعريف المنظار: تحوي لغة تعريف المعطيات في SQL تعليمة تسمح بتعريف منظار.
- السماحيات: تحوي لغة تعريف المعطيات في SQL تعليمات لتحديد حقوق الوصول إلى العلاقات والمنظائر.

- التكامل: تحوي DDL في SQL تعليمات لتحديد شروط التكامل.
- التحكم في المناقلا Transaction Control: في SQL يوجد تعليمات لتحديد بداية المناقلا ونهايتها. كما يوجد تطويرات تسمح بقفل المعطيات للتحكم في الوصول المتزامن.
- في الأمثلة المقبلة سنستخدم قاعدة معطيات حول المصارف التي يتضمن مخططها المنطقي:

```
Branch-Schema = (branch-name, branch-city, assets)
Customer = (customer-name, customer-street, city)
Loan = (branch-name, loan-number, amount)
Borrower = (customer-name, loan-number)
Account = (branch-name, account-number, balance)
Depositor = (customer-name, account-number)
```

2- لغة الاستعلام

2-1- البنية الأساسية للاستعلام في لغة SQL

تتألف البنية الأساسية للاستعلام في لغة SQL من ثلاثة أجزاء هي: Select, From, Where :

يُعبّر الجزء Select عن عملية الإسقاط في الجبر العلاقاقي. ويستخدم الجزء From لتحديد العلاقات المستخدمة في عملية الاختيار. ويكافئ الجداء الديكارتي في الجبر العلاقاقي. أما الجزء Where فيتضمن قضية منطقية يجب أن تحققها الواصفات الموجودة في العلاقات التي جرى تحديدها في جزء From.

ويُصبح الشكل العام للاستعلام في لغة SQL كما يلي :

```
select A1, A2, ..... An
from r1, r2, ..... rm
where P
```

حيث : A_i هي واصفات

r_i علاقات

P قضية

وهي تكافئ في الجبر العلاقتي العملية التالية :

$$\Pi_{A_1-A_n}(\sigma_P(r_1 \times r_2 \times \dots \times r_m))$$

سنناقش فيما يلي الأجزاء الثلاثة في استعلام بلغة SQL :

- فقرة الاختيار (Select clause) : تُستخدم لتحديد قائمة الواصفات المطلوب إظهارها في نتيجة الاستعلام.

مثال : لإيجاد أسماء جميع الأفرع في علاقة القروض، نستخدم التعليمة :

```
select branch-name
From loan;
```

والنتيجة هي علاقة تحوي واصفاً واحداً (branch-name)

تسمح لغة SQL بتكرار الحدوديات في العلاقة أو في نتيجة استعلام، ولحذف التكرار نضيف كلمة distinct بعد Select.

```
Select distinct branch-name
From loan;
```

وتُستخدم كلمة **all** لمنع حذف التكرار في الحدوديات.

مثال :

```
select all branch-name
from loan;
```

يستخدم الرمز "*" للدلالة على اختيار جميع الواصفات من علاقة .

مثال :

```
select *
From loan;
```

ويمكن أن تحوي فقرة **select** تعابير حسابية تستخدم العمليات + ، - ، * و /

مثال:

```
select branch-name, loan-number, amount * 100
from loan;
```

- فقرة **Where** : تُستخدم لتحديد الشروط التي يجب أن تحققها الحدوديات المختارة، فمثلاً لإيجاد أرقام القروض في الفرع "perryridge" والتي تزيد عن 1200 نكتب :

```
select loan-number
from loan
where branch-name = "perryridge" and amount >1200;
```

تستخدم المعاملات المنطقية and ، or و not في فقرة where وعمليات المقارنة < ، > ، <= ، >= و between و not between .

مثال :

Where amount between 90000 and 100000 ;

- فقرة From : تعرف فقرة From الجداء الديكارتي بين العلاقات المراد اختيار الحدوديات منها. ولما كانت عملية الدمج تُعرف كجداء ديكارتي وعملية اختيار وإسقاط، فإنه يمكننا التعبير بلغة SQL عن عملية الدمج التالية:

$$\Pi_{customer-name, loan-number}(borrower \times loan)$$

بالشكل:

```
select distinct customer-name, borrower. loan-number
From borrower, loan
where borrower.loan-number = loan.loan-number;
```

ويلاحظ في المثال السابق أننا استخدمنا relation-name.attribute-name في فقرة select بسبب وجود نفس اسم الواصف في أكثر من علاقة وذلك لتجنب الالتباس.

2-2- إعادة التسمية

تجري إعادة التسمية للعلاقات والواصفات باستخدام الفقرة as :

old-name as new-name

مثال : لإيجاد أسماء وأرقام قروض جميع الزبائن الذين حصلوا على قرض من فرع Perryridge ، مع التعويض عن اسم العمود loan-number باسم loan-id نكتب :

```
select distinct customer-name, borrower.loan-number as
loan-id
from borrower, loan
where borrower.loan-number = loan.loan-number and
branch-name= "Perryridge"
```

متحولات الحدودية :

عبارة "as" مفيدة في تعريف متحولات من نمط حدودية. وكما في لغة القضايا فإن المتحول الحدودي في SQL يرتبط بعلاقة. لنبين ذلك بالمثال التالي:

مثال : لإيجاد جميع الزبائن الذين حصلوا على قرض من المصرف (أسماء الزبائن وأرقام قروضهم)

```
Select distinct customer-name, T.loan-number
From borrower as T, loan as S
Where T.loan-number = S.loan-number
```

يلاحظ من المثال السابق أن المتحول الحدودي T يرمز إلى حدودية لا على التعيين من العلاقة borrower. وفي بعض الحالات نحتاج إلى إعطاء تسميات مختلفة لمتحولات حدودية. كما يوضح ذلك المثال التالي :

لنفترض أننا نحتاج إلى معرفة جميع الفروع التي توجد في نفس المدينة التي يوجد فيها الفرع Perryridge. لحل هذا الاستعلام نستخدم متحوليين الأول S والثاني T يشير كلاهما إلى العلاقة Branch-Schema.

Branch-name	Branch-city	Assets	
.			
perryridge			← S
.			
Y			← T

وُتستخدم التعليمة :

```
Select T.branch-name
From branch T,
Branch S
Where S.city = T.city and S.branch-name= "perryridge";
```

2-3- العمليات على سلسلة المحارف

أكثر العمليات استخداماً هي التشابه الجزئي like ونصف هنا التشابه باستخدام حرفين :

%. : للدلالة على أي سلسلة أحرف جزئية

under score : للدلالة على أي حرف

مثال: للبحث عن الفروع التي تحوي سلسلة الأحرف idge في أي موقع من اسم الفرع
نستخدم التعليمة :

```
Select branch-name From Branch
Where branch-name Like "%idge%";
```

وُستخدم "___" للدلالة على أي سلسلة أحرف مؤلفة من ثلاثة أحرف بالضبط ولوضع الحرفين %، ضمن السلسلة المراد مقارنتها، لا لاستخدام وظيفتهما. يجب أن يسبقا بحرف ESC. فنكتب:

```
customer-street like "abESC%cd%"
```

أي البحث عن اسم شارع سكن الزبون والذي يبدأ بسلسلة المحارف "ab%cd".

ويمكن استخدام التعبير not like لإيجاد عدم وجود تشابه ضمن سلسلة الأحرف. تسمح SQL باستخدام توابع مختلفة لسلسلة الأحرف مثل وصل سلسلتين Concatenating باستخدام ("II")، واستخراج جزء من السلسلة، وإيجاد طول سلسلة الأحرف، والتحويل بين الأحرف الصغيرة والكبيرة، ...

2-4- ترتيب الحدوديات الناتجة

يمكن للمستثمر أن يتحكم في ترتيب الحدوديات في العلاقة الناتجة، وذلك باستخدام عبارة Order by. فمثلاً لاستخراج قائمة مرتبة ترتيباً أبجدياً بأسماء الزبائن، والذين حصلوا على قرض من الفرع "Perryridge" نكتب:

```
select distinct customer-name
from borrower, loan
where borrower.loan-number = loan.loan-number
      and branch-name = "perryridge"
order by customer-name;
```

إن القائمة الناتجة ستكون مرتبة، ويمكن تحديد طريقة الترتيب تصاعدياً asc أو تنازلياً desc (descending ascending)، كما يمكن أن يطلب الترتيب على عدة واصفات.

مثال : لنفترض أننا نريد قائمة القروض مرتبة ترتيباً تنازلياً حسب مبلغ القرض، وفي حال وجود عدة قروض لها نفس المبلغ، نقوم بترتيبها تصاعدياً حسب رقم القرض.

نكتب:

```
select *
from loan
order by amount desc, loan-number asc
```

إن كلفة إجراء ترتيب على عدد كبير من الحدوديات هي كلفة عالية، ولذلك يجري إجراء الترتيب فقط في حال الضرورة.

2-5- العمليات على المجموعات

تسمح SQL-92 بالعمليات union , intersect و except المقابلة للاجتماع والتقاطع والفرق في الجبر العلاقتي. يجب أن تكون العلاقات التي تطبق عليها هذه العمليات متجانسة (لها نفس الواصفات).

سوف نبين كيفية استخدام هذه العمليات من خلال أمثلة :

2-5-1- الاجتماع

لإيجاد مجموعة كل الزبائن المتعاملين مع المصرف سواء مقترضين أو مودعين نكتب:

```
(select customer-name
From depositor)
union
```



```
select customer-name
from borrower)
```

تُحذف عملية الاجتماع الحدوديات المكررة آلياً مثل عملية الاختيار. في حال الرغبة في الاحتفاظ بالتكرار، نكتب (union all)، فتحتوي النتيجة حدوديات مكررة بعدد ظهورها في كلٍ من العلاقتين.

2-5-2- التقاطع

لإيجاد الزبائن الذين حصلوا على قرض ولديهم حساب في المصرف نكتب:

```
(select distinct customer-name
from depositor)
intersect
(select distinct customer-name
from borrower)
```

وتُستخدم العبارة intersect all لإظهار الحدوديات المكررة، وعندها تحتوي النتيجة الزبائن بعدد القروض والإيداعات المشتركة.

2-5-3- الفرق

لإظهار جميع الزبائن الذين لديهم حساب في المصرف ولم يستفيدوا من قرض منه نكتب:

```
(select distinct customer-name
from depositor)
except
(select customer-name
from borrower)
```

يمكن استخدام العبارة "except all" لإظهار الحدوديات المكررة، وبذلك نستطيع الحصول على أسماء الأشخاص الذين لديهم عدد من الحسابات أكبر من عدد القروض التي اقترضوها.

2-6- التوابع التجميعية

تُطبق التوابع التجميعية على مجموعة من القيم وتعيد قيمة واحدة. أهم هذه التوابع ما يلي :

التابع	دلالة التابع
Avg	المتوسط Average
Min	الأصغر Minimum
Max	الأكبر Maximum
Sum	المجموع Total

أمثلة :

• لإظهار متوسط أرصدة الحسابات في الفرع "x" نكتب :

```
select avg (balance)
from account
where branch-name = "x" ;
```

• وإظهار أكبر قيمة قرض منحها فرع Perryridge نكتب :

```
select Max (amount)
from loan
where branch-name= "Perryridge"
```

• وإظهار مجموع أرصدة الحسابات المفتوحة في الفرع "x" نكتب :

```
select Sum(balance)
from account
where branch-name ="x";
```

• لإظهار قائمة بالفرع ومتوسط الأرصدة في كل منها نكتب :

```
select branch-name, avg (balance)
from account
group by branch-name ;
```

نلاحظ من المثال السابق أن الإبقاء على التكرار ضروري أثناء عملية حساب الوسطي
.avg

2-7- معالجة القيم غير المعلومة

تسمح لغة SQL باستخدام القيمة غير المعلومة Null للدلالة على عدم توفر معلومات في
واصف.

مثال : لإيجاد أرقام القروض التي لا نعرف قيمتها نكتب:

```
select loan-number
from loan
where amount is null;
```

تعالج التوابع التجميعية القيم غير المعلومة بحيث تتجاهل وجود هذه القيم في مجموعة
العناصر التي تُجرى عليها العملية التجميعية وتعطي النتيجة دون اعتبارها، ويقوم
التابع Count بحصر عدد الحدوديات التي لا تحوي قيماً غير معلومة.

2-8- تجزئة العلاقة

في الأمثلة السابقة كنا نطبق تابعاً تجميعياً على بعض أسطر العلاقة التي تحقق شرطاً معيناً مثل : "اسم الفرع=..." غير أننا لو أردنا مثلاً حساب وسطي الأرصدة في كل فرع فإننا. وبحسب الطريقة السابقة، سنضطر لتنفيذ تعليمة :

```
Select Avg (balance)
from account
Where branch-name= "x";
```

عددًا من المرات يساوي عدد الفروع، وفي كل مرة نجعل x تأخذ اسم أحد الفروع. طبعاً هذا الحل غير مجدٍ وخاصة عندما لا تكون لدينا فكرة سابقة عن أسماء كافة الفروع. لحل هذه المشكلة، تقدم لغة SQL إمكان تجزئة العلاقة وفق قيمة أحد الواصفات، ومن ثم إجراء عملية تجميع على واحد أو أكثر من الواصفات الباقية.

نحدد العمود الذي تجري التجزئة وفقه بالعبارة Group by. وهكذا نكتب :

```
select branch-name, avg(balance)
from account
group by branch-name;
```

الذي يمكن تمثيله بالشكل التالي :

Account

Branch-name	Account-nb	Balance
Perryridge	X1	10000
National	X2	20000
Perryridge	X3	30000
Perryridge	X4	20000
Perryridge	X5	40000

Results

Branch-name	Avg (balance)
Perryridge	25000
National	20000

مثال :

« لإيجاد عدد المشتركين الذين لديهم حساب في كل فرع نكتب :

```
select branch-name, count (distinct customer-name)
from depositor, account
where depositor.account-number =
      account.account-number
group by branch-name ;
```

ويمكن وضع شروط تُطبق على نتيجة عملية التجميع باعتبارها علاقة جديدة.

مثال :

« لإيجاد أسماء الفروع التي متوسط الأرصدة فيها أكبر من \$1200 يُستخدم التعبير :

having في SQL كالتالي :

```
select branch-name, avg (balance)
from account
```

```
group by branch-name
having avg (balance) > 1200;
```

إذا وجد تعبير having و where في نفس الاستعلام فإن الشرط where يطبق أولاً، وتوضع الحدوديات المحققة للشرط في مجموعات (groups) بتطبيق group by ثم يطبق تعبير having على كل مجموعة. وتكون النتيجة المجموعات المستخدمة في تعليمة Select والتي تحقق الاستفسار.

مثال :

• لإيجاد متوسط أرصدة كل زبون يعيش في مدينة "x" ويملك على الأقل ثلاثة حسابات
نكتب :

```
select depositor, customer-name, avg (balance)
from depositor, account, customer
where depositor.account-number =
      account.account-number
and   depositor.customer-name =
      customer.customer-name
and customer-city = "x";
group by (depositor, customer-name)
having count (distinct depositor.account-number > = 3)
```

2-9- الاستفسارات الجزئية المضمنة

تقدم SQL تقانات لتنفيذ استفسارات جزئية مضمنة، الاستفسار الجزئي هو تعبير من الشكل Select -from -where ويكون مضمناً في استفسار آخر.

أمثلة :

الانتماء إلى مجموعة

لإيجاد جميع الزبائن الذين لديهم قرض وحساب في المصرف. يمكننا الوصول إلى المطلوب باستخدام عملية التقاطع بين مجموعتين: مجموعة الزبائن المقترضين و مجموعة الزبائن المودعين. كما يمكن استخدام منحنى آخر لإيجاد جميع المودعين في المصرف الذين ينتمون إلى مجموعة المقترضين من المصرف.

نكتب :

```
select distinct customer-name
from borrower
where customer-name in
      (select customer-name from depositor)
```

مقارنة المجموعات

تسمح لغة SQL باستخدام المقارنة بين مجموعة حدوديات ومجموعة حدوديات أخرى. مثال : لإيجاد أسماء جميع الفروع التي قيم موجوداتها أكبر من قيم موجودات فرع واحد على الأقل من الفروع الموجودة في مدينة "x". يمكن صياغة الاستعلام بالشكل التالي :

```
select distinct T.branch-name
from branch as T, branch as S
where T.assets > S.assets and S.branch-city = "x"
```

تسمح SQL بالصياغة بشكل مختلف وباستخدام عبارة بعض "some" التي تدل على بعض عناصر المجموعة، والعبارة all التي تدل على كل عناصر المجموعة. مثال :

```
select branch-name
from branch
where assets > some
      (select assets
       from branch
       where branch-city = "")
```

ويمكن استخدام عبارات المقارنة مع جزء من المجموعة التالية :

>some, <some, >=some, < >some, =some, <= some

وكذلك >all, <all, >=all, < >all, =all, <=all

مثال :

لإيجاد الفروع التي لديها متوسط الأرصدة أعظمي (أي متوسط الأرصدة فيها أكبر من جميع متوسطات الأرصدة في باقي الفروع) نكتب :

```
select branch-name
from account
group by branch-name
having avg (balance) >= all
      (select avg (balance)
       from account
       group by branch-name)
```

اختبار العلاقات الفارغة

يمكن اختبار وجود عناصر (حدوديات) في علاقة باستخدام عبارة يوجد "exists" التي تعيد قيمة صح "true" في حال وجود هذه العناصر.

مثال : لإيجاد أسماء الزبائن الذين لهم قروض وحساب في المصرف نكتب

```

select customer-name
from borrower
where exists (select *
              from depositor
              where depositor.customer-name =
                    borrower.customer-name)

```

اختبار عدم وجود تكرار في الحدوديات

يجري ذلك باستخدام التعبير "Unique" الذي يعيد القيمة true إذا لم يحتو الاستعلام الجزئي تكراراً في الحدوديات.

مثال: لإيجاد أسماء الزبائن الذين لديهم حساب واحد في الفرع "x" نكتب

```

select T.customer-name
from depositor as T
where unique (select R.customer-name
              from account, depositor as R
              where      T.customer-name = R.customer-name and
                        R.account-number = account.account-number
                        and
                        Account.branch-name = "x")

```

2-10- العلاقات المشتقة

يمكن حفظ نتيجة استفسار في علاقة جديدة. ويمكن إعادة تسمية واصفات هذه العلاقة الجديدة. وسنستخدم لذلك التعبير as :

مثال : لإنشاء علاقة اسمها result تحوي الوصفين branch-name و avg-balance

نكتب :

```
(select branch-name, avg (balance)
from account
group by branch-name)
as result (branch-name, avg-balance)
```

3- المناظير

يُعرّف المنظر بلغة SQL بالشكل:

```
create view as (query expression)
```

مثال:

```
create view branch-total-loan
(branch-name, total-loan)
as select branch-name, sum (amount)
from loan
group by branch-name
```

ولحذف منظر نكتب:

```
drop view view-name
```

4- تعديل قاعدة المعطيات

يجري التعبير عن عمليات الإضافة والحذف والتعديل على قاعدة المعطيات باستخدام

لغة SQL بالشكل التالي :

4-1- الحذف

الشكل العام لتعليمة حذف حدوديات من علاقة هو :

```
delete From tablename where condition
```

أمثلة :

1- لحذف جميع الحسابات العائدة للفرع "Perryridge" نكتب :

```
delete from account  
where branch-name= " Perryridge "
```

2- لحذف جميع الحسابات في جميع الفروع الموجودة في مدينة "Needham" نكتب :

```
delete from account  
where branch-name in (select branch-name  
from branch  
where branch-city= " Needham " )
```

3- لحذف كافة المودعين الذين فتحوا حسابات في فروع تقع في مدينة "Needham" نكتب :

```
delete from depositor  
where account-number in  
    (select account-number  
    from branch, account  
    where branch-city= " Needham "  
    and  
    branch.branch-name= account.branch-name  
    )
```

4- لحذف جميع تسجيلات الحسابات التي رصيدها أقل من وسطي الأرصدة في المصرف نكتب:

```
delete from account
where balance <
(select avg ( balance) from account)
```

المشكلة التي تظهر في هذا المثال هي أن حذف حدوديات من العلاقة account يغير من قيمة وسطي الأرصدة في المصرف، وهنا تقوم لغة SQL بحل هذا الإشكال كالتالي :

في البداية تقوم بحساب متوسط الأرصدة وإيجاد جميع الحدوديات التي يجب أن تحذف،

ثم تقوم بحذف جميع الحدوديات الموجودة سابقاً (دون إعادة حساب المتوسط أو إعادة اختبار الحدوديات).

4-2- الإضافة

تأخذ عملية إضافة حدودية إلى علاقة الشكل العام التالي :

```
insert into rel-name
values ( attribute values)
```

أو

```
insert into rel-name
(select ....From...Where ....)
```

وفي هذه الحالة يجري تنفيذ عملية الاختيار أولاً، ثم عملية الإضافة.

أمثلة :

1- لإضافة حدودية جديدة إلى علاقة الحسابات نكتب :

```
insert into account
values ( " Perryridge " , A-9732, 1200)
```

أو الشكل المكافئ :

```
insert into account
( branch-name, balance, account-number)
values
( " Perryridge " , 1200, A-9732)
```

2- لإضافة حدودية جديدة إلى علاقة الحسابات مع أن الرصيد للحساب مجهول أو غير

معلوم نكتب :

```
insert into account
values ( " Perryridge " , A-777, null)
```

3- لإضافة حساب مصرفي إلى جميع المقترضين من المصرف من الفرع "Perryridge" بقيمة \$200 وبحيث يُعتمد رقم القرض رقماً للحساب الجديد، نكتب :

```
insert into account
select branch-name, loan-number, 200
from loan
where branch-name= " Perryridge "
```

```
insert into depositor
select customer-name, loan-number
from loan, borrower
where branch-name= " Perryridge "
and loan.account-number= borrower.account-number
```

4-3- التعديل

الشكل العام لتعليمة التعديل هو :

```
update rel-name
set attribute= new-values
where condition
```

مثال : لتعديل أرصدة الحسابات المصرفية بإضافة 6% إلى الحسابات التي يزيد رصيدها عن على 10.000\$ وإضافة 5% إلى البقية، نكتب :

```
update account
set balance = balance * 1.06
where balance > 10000
```

```
update account
set balance = balance * 1.05
where balance <= 10000
```

ويمكن إجراء التعديل على قاعدة المعطيات بواسطة التعديل على المنظار المُعرف على هذه القاعدة. ولكن كما ناقشنا سابقاً (الفصل الثالث) تظهر بعض المشاكل المتعلقة بالتعامل مع القيم غير المعلومة.

مثال : ليكن لدينا المنظار branch-loan المُعرف على قاعدة معطيات المصرف، والذي يُظهر معطيات جميع القروض مع إخفاء المعطيات المتعلقة بكمية هذه القروض.

```
create view branch-loan as
select branch-name, loan-number
from loan
```

نضيف حدودية جديدة إلى المنظار :

```
insert into branch-loan
values ( " Perryridge " , " L-307 " )
```

تتمثل عملية الإضافة هذه بعملية إضافة للحدودية

("Perryridge " , " L-307 " , null)

إلى العلاقة loan.

التعديل على المعطيات من خلال مناهير أكثر تعقيداً ، وفي بعض الأحيان تستحيل ترجمته لمعرفة التعديلات الواجب إجراؤها على العلاقات الأساسية.

5- دمج العلاقات

دمج العلاقات عملية ثنائية تأخذ علاقتين كدخل وتُعطي علاقة كخرج. تُستخدم هذه العمليات الإضافية كتعابير استفسار جزئي في فقرة From ضمن الاستفسار.

شرط الدمج : يحدد الحدوديات المختارة من العلاقات والواصفات التي تظهر في نتيجة الدمج.

نوع الدمج : يحدد طريقة دمج أو عدم دمج الحدوديات من كل علاقة مع العلاقة الأخرى.

أنواع الدمج	شروط الدمج
دمج داخلي	طبيعي
دمج خارجي يساري	يحقّق <قضية>
دمج خارجي يميني	باستخدام (A1,A2,...,An)
دمج خارجي تام	

مثال : لتكن لدينا العلاقتان التاليتان :

العلاقة loan:

Branch-name	Loan-number	Amount
Downtown	L-170	3000
Redwood	L-230	4000
Perryridge	L 260	1700

العلاقة borrower

Customer-name	Loan-number
Jones	L-170
Smith	L-230
Hayes	L 155

تجري عملية الدمج الداخلي بين العلاقتين بالعملية التالية :

Loan inner join borrower on

loan.loan-number = borrower.loan-number

ونحصل على النتيجة :

Branch-name	Loan-number	Amount	Customer-name	Loan-number
Downtown	L-170	3000	Jones	L-170
Redwood	L-230	4000	Smith	L-230

وعملية الدمج الخارجي اليساري تُعطي :

loan left outer join borrower on

loan.loan-number = borrower.loan-number

Branch-name	Loan-number	Amount	Customer-name	Loan-number
Downtown	L-170	3000	Jones	L-170
Redwood	L-230	4000	Smith	L-230
Perryridge	L-260	1700	null	null

وعملية الدمج الطبيعي :

```
loan natural inner join borrower on
    loan.loan-number = borrower.loan-number
```

Branch-name	Loan-number	Amount	Customer-name
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith

وعملية الدمج الخارجي اليميني الطبيعي

```
loan natural right outer join borrower
```

Branch-name	Loan-number	Amount	Customer-name
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith
null	L-155	null	Hayes

وعملية الدمج الخارجي الكامل :

```
Loan full outer join borrower using ( loan-number )
```

Branch-name	Loan-number	Amount	Customer-name
Downtown	L-170	3000	Jones
Redwood	L-230	4000	Smith
Null	L-155	null	Hayes
Perryridge	L-260	1700	null

مثال : لإيجاد الزبائن الذين يتعاملون مع المصرف بنوع واحد من التعامل (مقترضون أو مودعون) لا بالنوعين معاً، نكتب :

```
select  customer-name
from    ( depositor natural full outer join borrower)
where   account-number is null
        or loan-number is null
```

6- لغة تعريف المعطيات DDL

تسمح لغة SQL بتعريف العلاقات التي تتكون منها قاعدة المعطيات ، ويتحديد معلومات عن كل علاقة بما يتضمن :

- المخطط العلاقتي لكل علاقة.
- مجال تعريف القيم المرتبطة بكل واصف.
- شروط التكامل.
- مجموعة المؤشرات المرتبطة بكل علاقة.
- أمن المعلومات وسماحيات الوصول.

- بنية التخزين الفيزيائي.

6-1- تعريف المجالات بلغة SQL

يمكن تعريف أنماط مختلفة للمجالات بلغة SQL منها :

char(n) : لتعريف سلسلة أحرف ذات طول ثابت. حيث n طول ثابت يحدده المستثمر.

varchar(n) : لتعريف سلسلة أحرف ذات طول متغير. حيث n الطول الأعظمي الذي يحدده المستثمر.

Int : عدد صحيح.

number(p,d) : رقم ممثل بالنقطة الثابتة.

real, double precision : رقم ممثل بالفاصلة العائمة مع دقة مضاعفة.

Date : التاريخ يحوي 4 خانات للسنة.

Time : الوقت في اليوم ، ساعة ، دقيقة ، ثانية.

ملاحظات :

- يُسمح باستخدام القيم غير المعلومة في جميع أنماط المجالات. والتصريح بكون واصفة لا تقبل قيماً غير معلومة يُحرم استخدام القيم غير المعلومة لهذه الواصفة.

- يمكن للمستثمر أن يُعرف مجالات خاصة به باستخدام لغة SQL كما في المثال التالي

```
create domain person-name char (20) not null
```

6-2- تعريف المخطط العلاقتي بلغة SQL

يُعرّف مخطط علاقة بلغة SQL باستخدام التعليمة التالية :

```
create table r ( A1 D1 ,A2 D2 , ..., An Dn ,
integrity-constraint l i ,
...,
integrity-constraint k i )
```

حيث :

r اسم العلاقة

Ai اسم الوصفة رقم I من مخطط العلاقة r

Di مجال تعريف الوصفة Ai

li, ...,ki شروط تكامل مُعرّف على الجدول

مثال :

```
create table branch
( branch-name char(15) not null ,
branch-city char(30),
assets integer)
```

تُعرّف شروط التكامل على المخطط العلاقتي والمتضمنة :

- عدم احتوائه على قيم غير معلومة not null.
- تعريف المفتاح الرئيسي (A1 ,...,An) primary key
- تعريف قضية (P) check حيث P قضية.

مثال :

لتعريف مخطط علاقتي branch بحيث تكون الوصفة branch-name مفتاحاً رئيسياً و قيم الوصفة assets غير سالبة، نكتب :

```
create table branch
(
branch-name char(15) not null,
branch-city char(30),
assets integer,
primary key ( branch-name), check ( assets>=0)
)
```

إن تعريف المفتاح الرئيسي على وصفة يجعل اختبار عدم احتوائها على قيم غير معلومة آلياً.

3-6- حذف مخطط علاقة بلغة SQL

تسمح تعليمة drop table بحذف جميع المعلومات المتعلقة بالعلاقة من قاعدة المعطيات، وتسمح تعليمة alter table بإضافة واصفات إلى مخطط علاقة موجودة، وبحيث تأخذ هذه الوصفة قيمة غير معلومة في جميع الحدوديات الموجودة سابقاً في العلاقة.

مثال : لإضافة واصف باسم A ومجاله D إلى العلاقة r نكتب :

```
alter table r add A D
```

ويمكن استخدام تعليمة alter table لحذف واصف من العلاقة. مثال :

```
alter table r drop A
```

حيث A واصف ضمن العلاقة r.

7- لغة SQL المضمنة

يمكن تضمين تعليمات SQL في لغات برمجة متعددة مثل : Pascal, PL/I, Fortran, C و Cobol. تُسمى اللغة المُضمنة تعليمات SQL باللغة المُضيف.

الشكل العام لتعليمات SQL المضمنة في لغة برمجة مثل PL/I هو :

```
EXEC SQL
    < embedded SQL statement >
END EXEC
```

مثال :

لإيجاد أسماء وأرقام الحسابات المصرفية للزبائن الذين يزيد رصيد حساباتهم عن amount ، نقوم بتحديد تعليمة SQL التي تسمح باستخراج تلك المعلومات وتعريف مؤشر cursor لتلك التعليمة كالتالي :

```
EXEC SQL
declare c cursor for
select customer-name, account-number
from depositor, account
where depositor.account-number= account.account-number
and account.balance> : amount
END-EXEC
```

- تسمح تعليمة open c حيث c مؤشر تعليمة SQL بوضع التعليمة موضع التنفيذ.

```
EXEC SQL
    open c
END-EXEC
```

- تسمح تعليمة fetch c into :variable-name بإسناد قيم واصفات حدودية واحدة من نتيجة الاستعلام إلى متحولات في اللغة المُضيف.

EXEC SQL

fetch c into :cn, :an

END-EXEC

ويُحصل على الحدوديات المختلفة الناتجة من استعلام معين، باستدعاء متتالٍ لتعليمة fetch، ويُستخدم متحول خاص لمنطقة اتصال لغة SQL للإشارة إلى الوصول إلى نهاية نتيجة الاستعلام.

– تسمح تعليمة close c لنظام قواعد المعطيات بحذف العلاقة الوسيطة الحاوية لنتيجة الاستعلام.

EXEC SQL

close c

END-EXEC

تمارين الفصل الرابع

1- أعدد كل طلبات المبينة في نهاية الفصل الثالث مع صياغة المطلوب بلغة SQL.

2- لتكن لدينا العلاقات التالية :

employee (Id, name, job, manager-id, salary, dept-id)

department (dept-id, name, city)

التي تتضمن معلومات عن الموظفين في شركة تمتلك عدة فروع.

حيث :

employee

Id	رقم الموظف
name	اسم الموظف
job	العمل الذي يؤديه الموظف
manager-id	رقم الرئيس المباشر
salary	الراتب الشهري
dept-id	رقم القسم الذي يعمل فيه

department

dept-id	رقم القسم
name	اسم القسم
city	المدينة التي يوجد فيها

المطلوب :

- أ. اكتب التعليمات اللازمة لإنشاء هاتين العلاقتين مع تحديد الشروط التالية :
 - رقم الموظف وحيد
 - لا يمكن أن يكون اسم الموظف مجهولاً
 - يعمل كل موظف في قسم واحد
- ب. هل يمكن اعتبار رقم الرئيس المباشر إجبارياً
- ت. هل يمكن اعتبار متضمناً في قائمة أرقام موظفي الشركة.
- ث. اكتب الاستفسارات التالية بلغة SQL :
 - عدد العاملين في مدينة دمشق مرتبة أبجدياً
 - أسماء ورواتب العاملين في قسم التسويق (Marketing) مرتبة تنازلياً وفق الراتب
 - اسم المدير العام للشركة
 - الموظفون الذين يتبعون مباشرة إلى المدير العام
 - الموظف (أو الموظفون) الذي يحصل على أعلى راتب في الشركة
 - أعداد العاملين في كل قسم من أقسام الشركة
 - اسم القسم الذي يحوي أكبر عدد من العاملين
 - اسم القسم الذي يزيد مجموع رواتب العاملين فيه عن 100000 ل. س
 - أسماء ورواتب العاملين الذين يزيد راتبهم عن راتب رئيسهم المباشر
 - القسم الذي يزيد متوسط رواتب العاملين عن متوسط رواتب العاملين في الشركة

الفصل الخامس

شروط التكامل

مقدمة

تُقدم شروط التكامل طرقاً للتوثيق من تناسق المعطيات وسلامة عمليات التعديل التي تُجرى على قاعدة المعطيات، كما تُقدم طرقاً لحماية قاعدة المعطيات من التخريب، وذلك بالتحقق أن التعديلات المسموح بها على قاعدة المعطيات لا تؤدي إلى الإخلال بتناسقها وصحتها.

يمكن أن تأخذ شروط التكامل، وفق ما عُرض سابقاً، أحد الأشكال التالية :

تعريف المفاتيح : حيث يجري تحديد مجموعة من الواصفات المكونة لمفتاح مرشح لصف كيانات، من ثم تصبح عملية الإضافة وعملية التعديل مشروطة بعدم وجود كيائين يشتركان في قيمة المفتاح المرشح.

نوع علاقة الارتباط : إن نوع الارتباط بين الكيانات : كثير-كثير، واحد-واحد، كثير-واحد، يضيف شروطاً تُفرض على الكيانات والارتباطات فيما بينها.

شرط التكامل هو قضية تُعبر عن تناسق المعطيات المخزنة في قاعدة المعطيات وصحتها. يمكن أن يكون اختبار تحقق القضية مكلفاً، ولذلك نقتصر عادة على تعريف شروط تكامل يمكن اختبارها بأقل كلفة.

1- تكامل المجالات

المجال بالتعريف هو مجموعة القيم الممكنة لوصف. تكامل المجال هو أحد أكثر أشكال شروط التكامل شيوعاً، التي تُختبر بسهولة لدى إدخال قيمة جديدة إلى قاعدة المعطيات. يمكن لأكثر من وصف أن يُعرف على المجال نفسه، ويقودنا تعريف تكامل المجال إلى اختبار القيم المدخلة إلى قاعدة المعطيات واختبار القيم المدخلة في الاستعلام للتوثق من إمكان المقارنة فيما بينهما.

تسمح تعليمة check في SQL-92 بقصر اختبار المجال على مجموعة من القيم، وهذا مما يسرع تنفيذ العملية.

مثال : لقصر مجال تعريف account-type على قيمتين فقط نكتب :

```
create domain account-type char(10)
constraint account-type-test
check( value in ("checking", "saving")
```

ولقصر مجال تعريف account-number بعدم احتوائه على null value نكتب :

```
Create domain account-number char(10)
constrait account-number-null-test
Check (value not null)
```

2- التكامل المرجعي

يمرّ التكامل المرجعي عن إمكان التوثق من تأثير قيم مجموعة من واصفات علاقة ما، في قيم مجموعة واصفات علاقة أخرى.

2-1- مفاهيم أساسية

الحدوديات السائبة **Dangling tuples** : ليكن لدينا زوج العلاقات (R, S) ، وليكن الدمج الطبيعي للعلاقين

$$R \times S$$

إن الحدوديات tr من العلاقة r والتي لا تندمج في حدوديات العلاقة s أي لا توجد حدوديات ts من s بحيث

$$tr[R \cap S] = ts[R \cap S]$$

تُسمى حدوديات سائبة.

يعتمد قبول مثل هذه الحدوديات على النمذجة المعتمدة لقاعدة المعطيات. ولقد رأينا سابقاً مفهوم الدمج الخارجي الذي يسمح بظهور مثل هذه الحدوديات.

مثال : لنأخذ المخطط العلاقتي لعلاقة الحسابات Account وعلاقة الفروع Branch. إن وجود حدودية مثل $t1$ من العلاقة account بحيث $t1[branch-name] = "x"$ وعدم احتواء العلاقة branch على حدودية t بحيث $t[branch-name] = "x"$ أمر مرفوض، فمن المفروض أن تحوي العلاقة branch جميع أسماء الفروع التابعة للمصرف. ومن ثم من المستحسن وجود شرط تكامل يمنع وجود مثل هذه الحدوديات السائبة.

بالطبع ليست جميع الحدوديات الطفيلة مرفوضة، فمثلاً وجود حدودية في العلاقة branch لا يمكن دمجها في العلاقة account لايسبب أية مشكلة إذا وجد فرع للمصرف لا يحوي حسابات مصرفية (في بداية إنشاء الفرع).

للتمييز بين هاتين الحالتين نستخدم ما يُسمى بالمفتاح الخارجي، وهو مجموعة الواصفات المضافة إلى العلاقة s والتي نرغب في عدم احتوائها على حدوديات سائبة بدمجها في العلاقة r وبحيث تُكوّن هذه الواصفات المفتاح الأولي للعلاقة r .

تعريف : ليكن لدينا علاقَتان $r_1(R_1)$, $r_2(R_2)$ لهما المفتاحان الأوليان K_1 , K_2 على الترتيب. نقول إن المجموعة الجزئية α من المخطط العلاقتي R_2 هي مفتاح خارجي يشير إلى K_1 من العلاقة r_1 إذا كان :

$$\forall t_2 \in r_2 \quad \exists t_1 \in r_1 \text{ where } t_1[K_1] = t_2[\alpha]$$

إن تحقق مثل هذا الطلب يسمى شرط تكامل مرجعي، أو ارتباط مجموعة جزئية.

إذا جرى اشتقاق المخطط العلاقتي من المخطط كيان-ارتباط، فإن العلاقات المشتقة من الارتباطات بين الكيانات تحوي شرط تكامل مرجعي. ثم إن العلاقة للمثابة لصف الكيانات الضعيف تحوي المفتاح الأولي لصف الكيانات القوي المرتبط به وهو يمثل المفتاح الخارجي الذي يقود إلى شرط تكامل مرجعي.

مثال : ليكن لدينا ارتباط R بين صفوف الكيانات E_1, E_2, \dots, E_n بحيث K_i هو مفتاح أولي لصف الكيانات E_i . إن العلاقة المثلة لـ R تحوي K_1, K_2, \dots, K_n حيث K_i هو مفتاح خارجي يقود إلى شرط تكامل مرجعي.

تعديل قاعدة المعطيات : سنبين طريقة تحقق التكامل المرجعي عند إجراء تعديل على قاعدة المعطيات. لتكن r_1, r_2 علاقَتين، تحوي r_2 مفتاحاً خارجياً مرتبطاً بالمفتاح الأولي r_1 ، ولنناقش عمليات الإضافة والحذف والتعديل على العلاقتين :

– الإضافة : حتى تحدث عملية إضافة حدودية t_2 إلى العلاقة r_2 يجب أن يقوم النظام بالتوثق من وجود t_1 في العلاقة r_1 بحيث يتحقق الشرط $t_1[K_1] = t_2[K_2]$ حيث K_1

المفتاح الأولي لـ $r1$ ، $K2$ المفتاح الخارجي لـ $r2$ المرتبط بـ $r1$. أما عملية إضافة حدودية إلى $r1$ فتجري دون أية حاجة لعملية اختبار.

– الحذف : لحذف حدودية $t1$ من $r1$ ، يقوم النظام بحساب الحدوديات من $r2$ المرتبطة بـ $t1$ وذلك بحساب التعبير الجبري التالي :

$$\sigma_{k2=t1[k1]}(r2)$$

وإذا كانت النتيجة مجموعة غير خالية تُجرى إحدى الحالتين التاليتين :

إعطاء رسالة خطأ وعدم تنفيذ عملية الحذف.

تُجرى عملية حذف لجميع الحدوديات التي تشير إلى $t1$ وهذا ما يُسمى بالحذف المتعاقب cascading.

– التعديل : سندرس عملية التعديل من خلال الحالتين التاليتين :

التعديل على العلاقة $r2$ الحاوية للمفتاح الخارجي.

التعديل على العلاقة $r1$ المرتبطة بـ $r2$ بواسطة مفتاحها الأولي.

إذا جرى تعديل الحدودية $t2$ من $r2$ وطال هذا التعديل قيمة المفتاح الخارجي $K2$ ، فإن النظام يجري اختباراً مشابهاً لعملية إضافة حدودية جديدة إلى $r2$.

وإذا جرى تعديل الحدودية $t1$ من $r1$ وطال هذا التعديل قيمة المفتاح الأولي $K1$ ، فإن النظام يجري عمليات مشابهة للعمليات المنفذة في حالة الحذف ، فيحسب مجموعة الحدوديات المحققة للشرط التالي :

$$\sigma_{k2=t1[k1]}(r2)$$

فإذا كانت هذه المجموعة غير خالية، قام النظام بإعطاء رسالة خطأ أو بإجراء عملية تعديل بالتسلسل على جميع الحدوديات المرتبطة بـ t1.

2-2- التكامل المرجعي في لغة SQL

يجري تحديد المفتاح الأولي والمفتاح الخارجي في SQL ضمن تعليمة بناء العلاقة create table حيث تُستخدم العبارات التالية :

Primary key لتحديد مجموعة الواصفات المكوّنة للمفتاح الأولي للعلاقة

Unique لتحديد قائمة الواصفات المكوّنة للمفتاح المرشح للعلاقة

Foreign key لتحديد قائمة الواصفات المكوّنة للمفتاح الخارجي.

أمثلة :

```
create table customer
( customer-name char(20) not null,
  customer-street char(30),
  customer-city char(30),
  primary-key (customer-name))

create table account
( account-number char(10) not null,
  branch-name char(15),
  balance integer,
  primary-key (account-number),
  foreign-key (branch-name)
  references branch (branch-name),
  check (balance >=0))
```



```

create table depositor
( customer-name char(20) not null,
  account-number char(10) not null,
  primary-key    (customer-name, account-number),
  foreign-key     (customer-name)
    references customer (customer-name),
  foreign-key     (account-number)
    references account (account-number)
    on delete cascade
    on update cascade)

```

3- التأكيد ضمن قاعدة المعطيات

التأكيد هو قضية تُعبر عن شرط نرغب في تحقيقه ضمن قاعدة المعطيات على الدوام. فشرط تكامل المجال وشرط التكامل المرجعي هما شكلان من أشكال التأكيد كُنّا قد ميزناهما بسهولة اختبارهما وسهولة تطبيقهما في تطبيقات قواعد المعطيات.

الشكل العام لتعريف شرط التأكيد بلغة SQL-92 هو :

```
create assertion <assertion-name> check <predicate>
```

مثال : لتعريف تأكيد في قاعدة المعطيات " مجموع القروض المأخوذة في كل فرع أقل من مجموع الأرصدة الموجودة في الفرع "

```

create assertion sum_constraint check
( not exists ( select *
               from branch
               where ( select sum(amount)
                       from loan
                       where loan.branch-name=
                           branch.branch-name

```

```

)
>=
( select sum(balance)
  from account
  where account.branch-name=,
        branch.branch-name
)
)

```

ملاحظة : بعد تعريف تأكيد في قاعدة المعطيات يقوم نظام إدارة قاعدة المعطيات بالتوثق من أن المعطيات المحتواة ضمن القاعدة تحققها، ومن ثم لا يسمح النظام بأي تعديل على القاعدة يؤدي إلى حدوث خلل في تحقيق التأكيد.

4- القادح Trigger

القادح مجموعة من التعليمات التي ينفذها آلياً نظام قواعد المعطيات لدى حدوث تعديل معين على القاعدة. يتحدد القادح بعنصرين أساسيين :

- الشروط التي سيجري تنفيذ القادح ضمنها،
- الوظائف التي ينفذها القادح.

مثال :

لنفترض أنه من غير المسموح به أن يأخذ رصيد حساب مصرفي قيمة سالبة، وأن من المستحسن عند إجراء عملية سحب من الحساب تحويل الرصيد إلى قيمة سالبة، تحويل الرصيد إلى الصفر وإعطاء قرض له نفس رقم الحساب المصرفي بقيمة تساوي الكمية الناقصة من الحساب. لتنفيذ هذا العمل يُعرف قادح بالشكل التالي :

شرط تنفيذ القادح هو حصول تعديل على قيمة الحساب جعلت قيمة الرصيد سالبة.

العمليات المضمنة في القادح هي :

- إضافة حدودية جديدة s في علاقة القروض loan بحيث يكون

```
s[branch-name]= t[branch-name]
s[loan-number]= t[account-number]
s[amount] = -t[balance]
```

- إضافة حدودية u إلى علاقة الاقتراض borrower بحيث

```
u[customer-name]= t[customer-name]
u[loan-number]= t[account-number]
```

- وضع $t[\text{balance}] = 0$

ويجري تعريف القادح بلغة SQL كما يلي :

```
define trigger overdraft on update of account T
(if new T.balance<0 then
  (insert into loan values
    (T.branch-name, T.account-number,
     - new T.balance)
  insert into borrower
  (select customer-name, account-number
   from depositor
   where T.account-number =
         depositor.account-number)
  update account s
  set s.balance =0
  where s.account-number= T.account-number))
```

5- الارتباطات التابعة Functional Dependencies

هي نوع خاص من شروط التكامل يعتبر تعميماً لمفهوم المفتاح.

5-1- مفاهيم أساسية

لقد عرفنا سابقاً مفهوم المفتاح الرئيسي لمخطط علاقتي $r(R)$ بأنه مجموعة الواصفات K التي تحقق الشرط : من أجل جميع الحدوديات t_1, t_2 من r و t_1 يختلف عن t_2 فإن $t_1[K]$ يختلف عن $t_2[K]$ أي لا توجد حدوديتان في r لهما نفس القيم لـ K . أما مفهوم الارتباط التابعي فهو تعميم لمفهوم المفتاح الرئيسي، فنقول إنه يوجد ارتباط تابعي بين α, β ونرمز إليه بـ $\alpha \rightarrow \beta$ من أجل

$$\alpha \subseteq R, \beta \subseteq R$$

إذا كان، في حال جميع العلاقات $r(R)$ ، جميع الحدوديات t_1, t_2 من r تُحقق :

$$t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$$

إن مفهوم الارتباط التابعي يسمح بالتعبير عن شروط تكامل لا يمكن التعبير عنها باستخدام مفهوم المفتاح.

مثال : لنأخذ المخطط العلاقتي التالي :

```
Loan-info-schema = ( branch-name,
                      loan-number,
                      customer-name,
                      amount)
```

مع الارتباطات التابعة التالية :

loan-number \rightarrow amount

loan-number \rightarrow branch-name

ولنفترض أنه يمكن أن يعطى القرض لأكثر من شخص (مثلاً حصول زوجين على قرض واحد) أي أنه لا يوجد ارتباط تابعي بين loan-number و customer-name. فالمفتاح الرئيسي للعلاقة هو (loan-number, customer-name) لا يكفي وحده للتعبير عن جميع الشروط الموجودة ضمن العلاقة ويُستخدم مفهوم الارتباط التابعي لتحديد شروط التكامل على عناصر العلاقة.

فإذا رغبتنا أن نتعامل مع المخطط العلاقتي R المحقق لمجموعة الارتباطات التابعة F، نقول إن العلاقة r محققة لـ F.

5-2- إغلاق مجموعة من الارتباطات التابعة

نقول عن ارتباط تابعي f إنه محقق منطقياً بتحقق الارتباطات التابعة F إذا استطعنا الوصول إليه بتطبيق مسلمات أرمسترونغ Armstrong والقواعد المشتقة منها. ونرمز إلى مجموعة الارتباطات التابعة التي تضم جميع الارتباطات التابعة المحققة منطقياً من F بـ F^+ وتُسمى بالمجموعة المغلقة لـ F.

تُستخدم الأحرف الإغريقية للدلالة على مجموعة الواصفات مثل: α, β ونرمز إليها بـ $\alpha\beta$ للدلالة على مجموعة اجتماع الواصفات، وتُستخدم الأحرف اللاتينية A, B, ... للدلالة على أسماء الواصفات.

مسلمات أرمسترونغ

الانعكاسية reflexivity : إذا كانت لدينا مجموعة من الواصفات

$$\alpha, \beta \text{ Where } \beta \subseteq \alpha \Rightarrow \alpha \rightarrow \beta$$

التزايد augmentation إذا كان $\alpha \rightarrow \beta$ ولتكن مجموعة من الواصفات γ فإن $\gamma\alpha \rightarrow \gamma\beta$.

قابلية التعدّي Transitivity

$$\alpha \rightarrow \beta \wedge \beta \rightarrow \gamma \Rightarrow \alpha \rightarrow \gamma$$

إضافةً إلى القواعد التالية التي يمكن البرهنة عليها باستخدام المسلمات السابقة :

$$\alpha \rightarrow \beta \wedge \beta \rightarrow \gamma \Rightarrow \alpha \rightarrow \beta\gamma \quad \text{الاجتماع :}$$

$$\alpha \rightarrow \beta\gamma \Rightarrow \alpha \rightarrow \gamma \wedge \alpha \rightarrow \gamma \quad \text{التجزئة :}$$

$$\alpha \rightarrow \beta \wedge \delta\beta \rightarrow \gamma \Rightarrow \alpha\delta \rightarrow \gamma \quad \text{لتعدّي الجزئي :}$$

مثال :

ليكن لدينا المخطط العلاقتي $R(A, B, C, G, H, I)$ والارتباطات التابعة F التالية :

A	→	B
A	→	C
CG	→	H
CG	→	I
B	→	H

والمطلوب :

إيجاد المجموعة المغلقة للارتباطات التابعة $F+$ ؟

تضم $F+$ مجموعة الارتباطات التابعة F إضافة إلى الارتباطات التابعة التالية :

$A \rightarrow H$ ناتج من تطبيق خاصية التعدي،

$CG \rightarrow HI$ ناتج من تطبيق خاصية الاجتماع للارتباطين

$CG \rightarrow I$ و $CG \rightarrow H$

$AG \rightarrow I$ لأن $AG \rightarrow C$ و $A \rightarrow I$ قاعدة التعدي الجزئي.

5-3- خوارزمية لإيجاد المجموعة المغلقة لمجموعة واصفات

لكي نحدد أ تكون مجموعة واصفات X مفتاحاً رئيسياً في علاقة ، يجب علينا حساب مجموعة الواصفات التي تتحدد تابعياً بـ X ونرمز إليها بـ $X+$ ، ويجري ذلك بتطبيق الخوارزمية التالية :

```

result := X;
while (changes to result) do
  for (each DF  $\alpha \rightarrow \beta$  in  $F$ ) do
    begin
      if  $\alpha \subseteq result$  then
        begin
           $result := result \cup \beta$ 
        end
    end
  end

```

5-4- التغطية الصغرى للارتباطات التابعة

يجب أن يقوم نظام قواعد المعطيات بالتأكد من بقاء جميع الارتباطات التابعة المعرفة على القاعدة صحيحة عند إجراء أي تعديل على قاعدة المعطيات، وعدم السماح بإجراء أي تعديل يسبب خللاً فيها. ولتقليص الجهد المطلوب لتحقيق ذلك، يجري الاختبار على مجموعة صغرى من الارتباطات التابعة، إغلاقها يساوي إغلاق مجموعة الارتباطات المعرفة على القاعدة.

تعريف :

نقول عن واصف A إنه واصف زائد في ارتباط تابعي $\alpha \rightarrow \beta$ ينتمي إلى مجموعة ارتباطات تابعة معرفة على قاعدة المعطيات إذا تمكنا من حذفه دون أن يؤثر ذلك في المجموعة المغلقة للارتباطات التابعة المعرفة على قاعدة المعطيات $F+$.

$$F+ = ((F - \{\alpha \rightarrow \beta\}) \cup \{\alpha - A \rightarrow \beta\}) +$$

نرمز إلى مجموعة التغطية الصغرى لـ F بـ F_c ويكون $(F_c)+ = F+$.

تحقق مجموعة التغطية الصغرى الخاصتين التاليتين :

- لا تحوي F_c أي ارتباط تابعي يتضمن واصفاً زائداً،
- لا تحوي ارتباطين تابعيين لهما القسم اليساري نفسه، فيجري تجميع الارتباطات ذات القسم اليساري المشترك في ارتباط تابعي وحيد.

تُحسب التغطية الصغرى لمجموعة ارتباطات تابعة بتطبيق الخطوات التالية :

1. تجميع الارتباطات التابعة ذات القسم اليساري المشترك في ارتباط تابعي واحد
(استخدام خاصية الاجتماع) أي يعوّض عن مجموعة الارتباطات التابعة من
الشكل :

$$\alpha 1 \rightarrow \beta 1$$

$$\alpha 1 \rightarrow \beta 2$$

$$\alpha 1 \rightarrow \beta 3$$

$$\alpha 1 \rightarrow \beta n$$

ب :

$$\alpha 1 \rightarrow \beta 1, \beta 2, \beta 3, \dots, \beta n$$

2. حذف الواصفات الإضافية إن وجدت في الارتباطات التابعة الناتجة F.

3. تكرار الخطوتين 1 و 2 حتى ثبوت F.

6- الخلاصة

يسمح تعريف شروط تكامل على قاعدة المعطيات بتأكيد تناسق المعلومات وصحتها وعدم ضياع هذا التناسق عند إجراء أي تعديل على قاعدة المعطيات.

عرضنا في هذا الفصل أنواعاً مختلفة لشروط التكامل، تناولت تعريف المفاتيح ونوع علاقات الارتباط، و شرط تكامل المجال، و شرط التكامل المرجعي، وتعريف التأكيد، وتعريف الارتباطات التابعة بين الواصفات.

تمارين الفصل الخامس

1- لتكن قاعدة المعطيات المعرفة في التمرين الثاني من الفصل السابق. بين أنواع وطريقة

تعريف شروط النكامل التالية :

- الراتب الأعلى الذي تدفعه هو 20000 ل. س.
- لا يمكن أن يزيد راتب أي عامل في قسم التسويق (Marketing) عن وسطي رواتب العاملين في قسم التصميم (Design)
- لا يمكن أن يزداد عدد العاملين في أي قسم من أقسام الشركة عن نصف العدد الكلي للعاملين في الشركة
- أسماء ورواتب العاملين الذين يزيد راتبهم عن راتب رئيسهم المباشر

2- أوجد التنظية الصغرى لمجموعة الارتباطات التابعة التالية المعرفة على المخطط

العلاقاتي (A,B,C) :

A	→	BC
B	→	C
A	→	B
AB	→	C

الفصل السادس

تصميم قاعدة معطيات علاقتية

مقدمة

يتطلب تصميم قاعدة معطيات علاقتية إيجاد مجموعة من مخططات علاقات، جيدة وقادرة على تمثيل جميع المعلومات المطلوبة بأقل تكرار ممكن. ويمكننا تلخيص أهداف التصميم بثلاث نقاط وهي :

- إنقاص ظاهرة تكرار المعطيات ما أمكن،
 - التوثيق من تمثيل العلاقات الموجودة بين الواصفات،
 - سهولة اختبار التعديلات لاختبار تحقق شروط التكامل المعرفة على قاعدة المعطيات.
- لنورد بعض المشاكل الممكن ظهورها أثناء تصميم قاعدة المعطيات، من خلال دراسة المثال التالي :

ليكن لدينا المخطط العلاقتي :

Lending-schema = (branch-name, branch-city, assets,
customer-name, loan-number, amount)

نلاحظ ما يلي :

إن المعطيات حول (branch-name, branch-city, assets) مكررة في حال كل قرض يقرضه فرع (مشكلة تكرار في المعطيات)، إن هذا التكرار يقودنا إلى القيام بتعديل عدة حدوديات في العلاقة إذا أردنا تعديل معلومة واحدة مثلاً : اسم الفرع.

عدم إمكان تخزين معلومات عن فرع دون وجود قرض على الأقل يقرضه هذا الفرع (عدم القدرة على تمثيل جميع المعلومات). أحد الحلول الممكنة في هذه الحالة هو استخدام القيمة null (عدم التعين لقيمة) وإضافة الحدودية المرادة إلى العلاقة بتكملة المعلومات المطلوبة وغير المتوفرة بهذه القيمة.

بالإمكان التفكير في حل هذه المشاكل عن طريق تجزئة المخطط العلاقتي، ذي العدد الكبير من الواصفات، إلى مجموعة من المخططات العلاقتية ذات حجم أقل من الواصفات، ولكن يمكن أن نقودنا تجزئة غير مدروسة إلى أشكال أخرى من المشاكل. سنوضح ذلك بدراسة اقتراح تجزئة المخطط العلاقتي السابق إلى المخططين العلاقتيين التاليين :

```
Branch-customer-schema =(branch-name, branch-city,
                           assets, customer-name)
Customer-loan-schema =(customer-name, loan-number,
                        amount)
```

نلاحظ أن العلاقتين الناتجتين عن التجزئة لا تحويان معلومات عن الفروع والقروض المأخوذة منها، إذ يمكن لزبون أن يقرض من أكثر من فرع. وإذا أجرينا عملية دمج للعلاقتين السابقتين نحصل على حدوديات غير موجودة أصلاً في العلاقة الأصلية، أي يوجد ضياع في المعلومات، ونقول إن التجزئة تحوي ضياعاً بالمعلومات Lossy-join-decomposition وهي أحد المشاكل في تصميم قاعدة المعطيات بعد التجزئة.

وبوجه عام يجب على التجزئة المقترحة أن تحقق مجموعة من الخواص لتكون تصميماً جيداً لقاعدة المعطيات وهي :

جميع الواصفات الموجودة في المخطط العلاقتي الأصلي R موجودة في المخططين العلاقتيين الجزأين R1, R2 .

$$R = R_1 \cup R_2$$

التجزئة محافظة على المعطيات Lossless-join decomposition أي لا تحوي ضياعاً في المعلومات ، نقول إن التجزئة محافظة على المعطيات إذا تحقق الشرط التالي :

مهما يكن r معرفة على المخطط العلاقتي R فإن التجزئة R_1, R_2 تحقق

$$r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$$

مثال :

لنأخذ التجزئة غير المحافظة على المعطيات التالية للمخطط العلاقتي $R(A, B)$:

$$R_1 = (A) \quad R_2 = (B)$$

A	B
α	1
α	2
β	1

r

A
α
β

$\Pi_A(r)$

B
1
2

$\Pi_B(r)$

• $\Pi_A(r) \bowtie \Pi_B(r)$

A	B
α	1
α	2
β	1
β	2

1- التقييس باستخدام الارتباطات التابعة

الهدف الذي نرغب في تحقيقه أثناء تصميم قاعدة معطيات هو الابتعاد عن الخواص غير المرغوب فيها (التكرار، عدم القدرة على تمثيل بعض المعلومات)، ولذلك نقوم بتجزئة العلاقات الكبيرة إلى علاقات أصغر منها بحيث تكون التجزئة محافظة على المعطيات، كما نقوم باستخدام الارتباطات التابعة، التي رأينا كيفية استخدامها لتعريف شروط تكامل على المعطيات، لتعريف أشكال نظامية للعلاقات تكون تصاميم جيدة لقاعدة معطيات.

1-1- خواص التجزئة المستخدمة للتقييس

تجزئة محافظة على المعطيات **Lossless-join decomposition** :

ليكن لدينا المخطط العلاقتي R و F مجموعة الارتباطات التابعة المعرفة على R . وليكن R_1, R_2 شكلين من أشكال التجزئة لـ R . نقول إن التجزئة هي محافظة على المعطيات إذا وجد على الأقل أحد الارتباطات التالية ضمن F :

$$R_1 \cap R_2 \rightarrow R_1$$

$$R_1 \cap R_2 \rightarrow R_2$$

تجزئة محافظة على الارتباطات التابعة **Dependency perservation** :

عندما تبقى مجموعة شروط التكامل (الارتباطات التابعة) المعرفة على المخطط العلاقتي صحيحة بعد تجزئته، نقول إن التجزئة محافظة على الارتباطات. ويجري اختبار ذلك كما يلي :

ليكن R المخطط العلاقتي الذي قمنا بتجزئته إلى R_1, R_2, \dots, R_n

ليكن F مجموعة الارتباطات التابعة المعرفة على R

ليكن F_i مجموعة الارتباطات التابعة من F^+ للواصفات الموجودة في R_i .

نختبر العلاقة التالية، وفي حال تحققها نقول إن التجزئة محافظة على الارتباطات

$$(F_1 \cup F_2 \cup \dots \cup F_n)^+ = F^+$$

التابعة.

No redundancy عدم تكرار المعطيات

من الممكن أن لا نستطيع حذف ظاهرة تكرار المعطيات بعملية التجزئة، ولكن المرغوب فيه هو تقليص هذه الظاهرة قدر الإمكان.

1-2- المنهجية المتبعة في التقييس

نقوم بدراسة المخططات العلاقتية المقترحة لقاعدة المعطيات وإقرار كون المخطط العلاقتي جيداً. وإذا لم يكن المخطط ليس جيداً، يُجزأ إلى مجموعة علاقات $\{R_1, R_1, \dots, R_n\}$ بحيث تكون كلٍ من هذه العلاقات جيدة والتجزئة المقترحة محافظة على المعطيات ومحافظة على الارتباطات التابعة المعرفة على المخطط العلاقتي الأصلي.

2- الأشكال النظامية

1-2 - الشكل النظامي الأول First Normal Form 1NF

نقول عن علاقة إنها من الشكل النظامي الأول إذا كان تقاطع (سطر، عمود) يحوي قيمة واحدة.

مثال : لتكن العلاقة PRODUCTS

PRODUCTS	Factory	Products
	F1	{p2,p4,p5}
	F2	{p1, p2,p3}

هذه العلاقة ليست من الشكل النظامي الأول وتُصبح من الشكل النظامي الأول في حال

كان محتواها كالتالي :

PRODUCTS	Factory	Product
	F1	p2
	F1	p4
	F1	p5
	F2	p1
	F2	p2
	F2	P3

2-2- الشكل النظامي الثاني 2NF Second Normal Form

تعريف :

الارتباط التابعي : نقول إنه يوجد ارتباط تابعي بين X و Y إذا تحقق الشرط التالي :

$$X \rightarrow Y \Rightarrow (\forall t1, t2 \in r, t1[X] = t2[X] \Rightarrow t1[Y] = t2[Y])$$

المفتاح : ليكن المخطط العلاقتي $R(A_1, A_2, \dots, A_n)$ وليكن X مجموعة من واصفات هذا المخطط، نقول إن X تُكوّن مفتاحاً للعلاقة R إذا تحقق الشرطان التاليان :

– مهما تكن Y مجموعة من الواصفات المنتمجة إلى R فإنه يوجد ارتباط تابعي

$$\forall Y \in R \Rightarrow X \rightarrow Y$$

من X إلى Y . أي :

– لا توجد مجموعة جزئية من X تحقق الشرط 1.

الواصفات الأولية : تُسمي الواصفات المنتمجة إلى أحد مفاتيح العلاقة بالواصفات الأولية.

الارتباط التام : نقول إنه يوجد ارتباط تابعي تام بين X و Y إذا وفقط إذا تحقق الشرطان التاليان :

– يوجد ارتباط تابعي بين X, Y

– لا يوجد مجموعة جزئية من X تربط بارتباط تابعي بـ Y .

الشكل النظامي الثاني : نقول عن علاقة إنها من الشكل النظامي الثاني إذا كانت من الشكل النظامي الأول وكانت جميع الواصفات غير الأولية فيها مرتبطة ارتباطاً تابعياً تاماً بكافة مفاتيح العلاقة.

2-3 – الشكل النظامي BCNF (Boyce-Codd Normal Form)

نقول إن العلاقة R من الشكل النظامي BCNF إذا كانت جميع الارتباطات التابعة المعرفة على العلاقة F^+ وليكن الارتباط التابعي من X نحو Y هو أحد الأنواع التالية :

- ارتباط تابعي بديهي، ونقول عن ارتباط تابعي من X نحو Y إنه بديهي إذا كانت Y محتواة في X.

- X تكون مفتاحاً رئيسياً للعلاقة R.

إن الشكل النظامي BCNF هو أحد الأشكال النظامية المرغوب في تحقيقها في تصميم قاعدة المعطيات. ونقول إن تصميم قاعدة المعطيات هو من نوع BCNF إذا كانت جميع المخططات العلاقاتية المؤلفة للقاعدة هي من الشكل BCNF.

مثال :

ليكن $R = (A, B, C)$ والارتباطات التابعة المعرفة على R هي :

$$A \rightarrow B$$

$$B \rightarrow C$$

وليكن مفتاح العلاقة هو A.

إن العلاقة R ليست من الشكل النظامي BCNF لوجود الارتباط التابعي الثاني. إن تجزئتها إلى العلاقتين $R_1 = (A, B)$, $R_2 = (B, C)$ لها الخواص التالية :

- R_1, R_2 من الشكل النظامي BCNF.

- التجزئة محافظة على المعطيات.

- التجزئة محافظة على الارتباطات التابعة.

2-3-1 خوارزمية التجزئة

نبين فيمايلي خوارزمية لتجزئة مخطط علاقاتي ليس من الشكل النظامي BCNF إلى مجموعة علاقت من الشكل النظامي BCNF.

```

result:= { R } ;
done:= false;
compute F + ;
while (not done) do
    if (there is a schema Ri in result
        that is not in BCNF) then
        begin
            let  $X \rightarrow Y$  be a nontrivial functional
            dependency that holds on Ri such that
             $X \rightarrow Ri$  is not in F + ,
            and  $X \cup Y = 0$ 
            result := (result-Ri) U (Ri-Y) U (X,Y);
        end
    else done:= true;

```

ملاحظة :

إن التجزئة الناتجة هي من الشكل BCNF وليست بالضرورة محافظة على المعطيات.

مثال :

ليكن لدينا المخطط العلاقتي التالي :

$R = (\text{branch-name}, \text{branch-city}, \text{assets}, \text{customer-name},$

loan-number, amount)

ونتكن الارتباطات التابعة التالية المعرفة على المخطط :

$F = \{ \text{branch-name} \rightarrow \text{assets branch-city}$
 $\text{loan-number} \rightarrow \text{amount branch-name}$

Key = { loan-number, customer-name } : مفتاح هو :

بتطبيق خوارزمية التجزئة نحصل على المخططات العلاقتية التالية :

$R1 = (\text{branch-name, branch-city, assets})$

$R2 = (\text{branch-name, customer-name, loan-number, amount})$

$R1$ من الشكل النظامي BCNF و $R2$ ليس من الشكل النظامي BCNF ولذلك نقوم

بتجزئته إلى العلاقتين التاليتين :

$R3 = (\text{branch-name, loan-number, amount})$

$R4 = (\text{customer-name, loan-number})$

وتكون التجزئة النهائية للمخطط العلاقتي R هي :

$R1, R3, R4$

ملاحظة :

ليس بالإمكان دوماً الحصول على تجزئة من الشكل النظامي BCNF تكون محافظة على الارتباطات التابعة المعرفة على المخطط العلاقتي الأصلي.

مثال :

ليكن لدينا المخطط العلاقتي $R = (J, K, L)$ والارتباطات التابعة المعرفة عليه :

$F = \{ JK \rightarrow L, L \rightarrow K \}$

للعلاقة مفتاحان مرشحان هما : (J, K) و (J, L)

R ليست من الشكل النظامي BCNF وأي تجزئة لها من الشكل BCNF ستفشل بالمحافظة على الارتباط التابعي $JK \rightarrow L$.

4-2 – الشكل النظامي الثالث (Third Normal Form)

نقول عن مخطط علاقتي R إنه من الشكل النظامي الثالث إذا كانت جميع الارتباطات التابعة من الشكل $X \rightarrow Y$ المنتمية إلى المجموعة المغلقة للارتباطات التابعة المعرفة على هذا المخطط لها أحد الأشكال التالية :

الارتباط التابعي $X \rightarrow Y$ هو ارتباط تابعي بديهي (Y محتوى ضمن X).

X هو مفتاح رئيسي في R.

كل الواصفات A المنتمية إلى مجموعة الواصفات Y-X تنتمي إلى أحد المفاتيح المرشحة للمخطط العلاقتي.

ملاحظة :

كل علاقة من الشكل النظامي BCNF هي من الشكل النظامي 3NF (ذلك أن الارتباطات التابعة المعرفة عليها لها أحد الشكلين الأوليين من الأشكال الثلاثة السابقة الموجودة في تعريف الشكل النظامي الثالث).

مثال :

ليكن لدينا المخطط العلاقتي التالي مع الارتباطات التابعة المعرفة عليه :

$$R = (J, K, L)$$

$$F = \{ JK \rightarrow L, L \rightarrow K \}$$

المفتاحان المرشحان للعلاقة هما (J, K) و (J, L).

العلاقة R هي من الشكل النظامي الثالث حيث

$JK \rightarrow L$ JK هي مفتاح رئيسي

$L \rightarrow K$ K محتوى في مفتاح مرشح.

2-4-1 خوارزمية التجزئة

نبين فيما يلي خوارزمية تجزئة مخطط علاقتي R إلى مجموعة من المخططات العلاقتية $\{R_1, R_2, \dots, R_n\}$ بحيث كل مخطط علاقتي R_i هو من الشكل 3NF، والتجزئة محافظة على المعطيات، والتجزئة محافظة على الارتباطات التابعة.

ليكن لدينا F_c التغطية الصغرى لمجموعة الارتباطات التابعة F المعرفة على المخطط العلاقتي R.

```
i := 0;
for each functional dependency  $X \rightarrow Y$  do
  if none of the schemas  $R_j$  ,  $1 \leq j \leq i$  contains XY
  then begin
    i := i + 1;
     $R_i := (X, Y)$  ;
  end
  if none of the schemas  $R_j$  ,  $1 \leq j \leq i$  contains
  a candidate key for R then
    begin
      i := i + 1;
       $R_i := \text{any candidate key for R}$ ;
    end
return (  $R_1, R_2, \dots, R_i$  )
```

مثال :

ليكن لدينا المخطط العلاقتي المتعلق بالمصارف التالي :

Banker-info-schema = (branch-name, customer-name,
banker-name, office-number)

والارتباطات التابعة المعرفة عليه :

banker-name → branch-name office-number

customer-name branch-name → banker-name

المفتاح المرشح للمخطط هو :

{customer-name, branch-name }

وبتطبيق خوارزمية التجزئة نجد المخططات التالية :

Banker-office-schema =(banker-name, branch-name,
office-number)

Banker-schema = (customer-name, branch-name,
banker-name)

ملاحظة :

نجد أنه بالإمكان دوماً تجزئة علاقة إلى مجموعة علاقات من الشكل 3NF بحيث تكون التجزئة محافظة على المعطيات ومحافظة على الارتباطات التابعة. كما أنه بالإمكان دوماً تجزئة علاقة إلى مجموعة علاقات من الشكل BCNF بحيث تكون التجزئة

محافظة على المعطيات ولكن يمكن أن يكون من المستحيل الحصول على تجزئة محافظة على الارتباطات التابعة.

الهدف المرجو تحقيقه في تصميم قاعدة معطيات علاقاتية هو الوصول إلى مجموعة مخططات علاقاتية تكون :

- من الشكل BCNF،
 - محافظة على المعطيات
 - محافظة على الارتباطات التابعة.
- وفي حال استحالة تحقيق ذلك، يمكن القبول بتصميم يحقق :
- المخططات العلاقاتية من الشكل 3NF،
 - المحافظة على المعطيات
 - المحافظة على الارتباطات التابعة.

2-5 - الشكل النظامي الرابع (Fourth Normal Form)

يمكن أن تكون مخططات علاقاتية لقاعدة المعطيات من الشكل BCNF وتبدو أنها ليست نظامية بقدر كافي. مثال على ذلك قاعدة المعطيات تحوي المخطط العلاقتي :

classes (course, teacher, book)

حيث تعني الحدودية (c,t,b) أن المعلم t مؤهل لتدريس المادة c وأن الكتاب التدريسي للمادة c هو b.

تضم قاعدة المعطيات قائمة بالمدربين المؤهلين لتدريس كل مادة من المواد، ومجموعة الكتب التدريسية اللازمة لكل مادة (لا يوجد معلومات عن الذين يدرسون هذه الكتب التدريسية).

Course	Teacher	book
Database	Avi	Korth
Database	Avi	Ullman
Database	Hank	Korth
Database	Sudarshan	Ullman
Database	Sudarshan	Korth
Operating systems	Avi	Silberschatz
Operating systems	Avi	Shaw
Operating systems	Jim	Silberschatz
Operating systems	Jim	Shaw

classes

إن المفتاح المرشح للعلاقة هو {course, teacher, book} ولا يوجد ارتباط تابعي غير بديهي ضمن العلاقة، من ثم المخطط العلاقتي هو من الشكل BCNF.

نلاحظ أنه عندما نريد أن نضيف معلومة وصول مدرس جديد Sara مؤهل لتدريس مادة database نحتاج إلى إضافة حدوديتين في العلاقة وهما :

(database, Sara, Korth)

(database, Sara, Ullman)

وهذا يشعرنا بوجود مشكلة مع أن المخطط هو من الشكل BCNF ومن المفضل تجزئته إلى مخططين هما :

Course	Teacher
Database	Avi
Database	Hank
Database	Sudarshan
Operating systems	Avi
Operating systems	Jim

Teaches

Course	Book
Database	Korth
Database	Ullman
Operating systems	Silberschatz
Operating systems	Shaw

Text

سنرى أن المخططين الجديدين هما من الشكل النظامي الرابع.

2-5-1- الارتباط المتعدد القيم Multivalued Dependencies

ليكن لدينا المخطط العلاقتي R و X, Y مجموعة واصفات محتواة في المخطط العلاقتي R ، نقول إنه يوجد ارتباط متعدد القيم $X \twoheadrightarrow Y$ إذا تحقق الشرط التالي من أجل جميع العلاقات المعرفة على R ولتكن $r(R)$:

إن وجدت حدوديات t_1, t_2 من r بحيث $t_1[X] = t_2[X]$ فإنه يوجد حدوديتان t_3, t_4 من r بحيث يتحقق ما يلي :

$$t_1[X] = t_2[X] = t_3[X] = t_4[X]$$

$$t_3[Y] = t_1[Y]$$

$$t3[R-Y] = t2[R-Y]$$

$$t4[Y] = t2[Y]$$

$$t4[R-Y] = t1[R-Y]$$

يمكن تمثيل ذلك بالجدول :

	X	Y	R-X-Y
T1	a1a2.....ai	ai+1.....aj	aj+1.....an
T2	a1a2.....ai	bi+1.....bj	bj+1.....bn
T3	a1a2.....ai	ai+1.....aj	bj+1.....bn
T4	a1a2.....ai	bi+1.....bj	aj+1.....an

مثال :

ضمن مثالنا قاعدة معطيات classes نلاحظ وجود ارتباطين تابعيين متعددي القيم هما :

course →→ teacher

course →→ book

نلاحظ أنه إذا وجد ارتباط تابعي بين $X \rightarrow Y$ فإنه يوجد ارتباط تابعي متعدد القيم بين $X \rightarrow \rightarrow Y$.

2-5-2- نظرية الارتباطات المتعددة القيم

ليكن لدينا D مجموعة الارتباطات التابعة والارتباطات المتعددة القيم، نسمي D^+ المجموعة المغلقة لـ D جميع الارتباطات التابعة المنبثقة منطقياً من D، والتي يمكن حسابها بتطبيق القواعد والمسلمات التالية :

الانعكاسية : إذا كان لدينا X, Y مجموعة من الواصفات و Y محتواة ضمن X فإن $X \rightarrow Y$ محقق.

التزايد : إذا كان $X \rightarrow Y$ محققا و Z مجموعة من الواصفات ، فيكون $ZX \rightarrow ZY$ محققا.

التعدي : إذا كان $X \rightarrow Y$ محققا و $Y \rightarrow Z$ محققا فإن $X \rightarrow Z$ محقق.

الإتمام : إذا كان $X \rightarrow Y$ محققا فإن $X \rightarrow \rightarrow R-X-Y$ محقق.

التزايدية المتعددة القيم : إذا كان $X \rightarrow \rightarrow Y$ محققا و Z مجموعة من الواصفات من R ،

و M مجموعة واصفات محتواة في Z ، فإن $ZX \rightarrow \rightarrow MY$ محقق.

المطابقة : إذا كان $X \rightarrow Y$ محققا فإن $X \rightarrow \rightarrow Y$ محقق.

الالتحام : إذا كان $X \rightarrow \rightarrow Y$ و Z محتواة في Y و $M \text{ intersect } Y$ يساوي خالية و

$M \rightarrow Z$ فإن $X \rightarrow Z$ محقق. ... =

التزايدية المتعددة القيم : إذا كان $X \rightarrow \rightarrow Y$ و $X \rightarrow \rightarrow Z$ فإن $X \rightarrow \rightarrow YZ$.

التقاطع : إذا كان $X \rightarrow \rightarrow Y$ و $X \rightarrow \rightarrow Z$ فإن $X \rightarrow \rightarrow Y \cap Z$ محقق.

الفرق : إذا كان $X \rightarrow \rightarrow Y$ و $X \rightarrow \rightarrow Z$ فإن $X \rightarrow \rightarrow Y - Z$ محقق و $X \rightarrow \rightarrow Z - Y$ محقق.

محقق.

مثال :

ليكن لدينا المخطط العلاقتي $R(A,B,C,A,H,I)$ والارتباطات التابعة المعرفة عليه :

$D = \{ A \rightarrow \rightarrow B, B \rightarrow \rightarrow HI, CG \rightarrow H \}$ والمطلوب حساب D^+ .

2-5-3- الشكل النظامي الرابع

نقول عن مخطط علاقتي إنه من الشكل النظامي الرابع إذا كانت جميع الارتباطات التابعة المتعددة القيم المعرفة عليه، والتي هي من الشكل $X \rightarrow Y$ تحقق على الأقل أحد الشرطين التاليين :

• $X \rightarrow Y$ هو ارتباط تابعي متعدد القيم بديهي .

• X هو مفتاح رئيسي للمخطط العلاقتي R .

ملاحظة :

كل مخطط علاقتي من الشكل النظامي الرابع هو من الشكل النظامي BCNF.

2-5-4- خوارزمية التجزئة

فيما يلي خوارزمية لتجزئة مخطط علاقتي إلى مجموعة مخططات من الشكل النظامي الرابع

```
result:= { R } ;
done:= false;
compute F + ;
while ( not done) do
if (there is a schema Ri in result that is not in 4NF)
then
begin
```

ليكن لدينا $X \rightarrow Y$ ارتباط تابعي متعدد القيم غير بديهي

محقق على R_i حيث $X \rightarrow R_i$ ليست ضمن مجموعة الارتباطات التابعة

```

        .  $Y \rightarrow X = \emptyset$  و  $F+$ 
        result := (result -  $R_i$ )  $\cup$  ( $R_i - Y$ )  $\cup$  ( $X, Y$ );
    end
else done := true;

```

ملاحظة :

التجزئة الناتجة من الشكل النظامي الرابع محافظة على المعطيات.

مثال :

ليكن لدينا المخطط العلاقتي $R(A, B, C, A, H, I)$ والارتباطات التابعة المعرفة عليه :

$D = \{ A \rightarrow B, B \rightarrow HI, CG \rightarrow H \}$ والمطلوب إيجاد التجزئة من الشكل النظامي الرابع

- نلاحظ أن R ليست من الشكل النظامي الرابع لأن $A \rightarrow B$ و A ليست مفتاحاً رئيسياً للعلاقة.

- بتطبيق خوارزمية التجزئة نحصل على :

- a) $R_1 = (A, B)$ (R_1 is in 4NF)
 b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF)

ولأن $CG \rightarrow H$ محقق نحصل على تجزئة R_2 إلى :

- c) $R_3 = (C, ?, H)$ (R_3 is in 4NF)
 d) $R_4 = (A, C, ?, I)$ (R_4 is not in 4NF)

- ولما كان $A \rightarrow B, B \rightarrow HI, A \rightarrow I$ فإن $A \rightarrow HI$ ويجزأ R_4 إلى :

- e) $R_5 = (A, I)$ (R_5 is in 4NF)
 f) $R_6 = (A, C, ?)$ (R_6 is in 4NF)

2-5-5- المحافظة على الارتباطات المتعددة القيم

ليكن لدينا المخططات العلاقتية $R_1, R_2, R_3, \dots, R_n$ التي هي تجزئة للمخطط العلاقتي R ، وليكن D مجموعة الارتباطات التابعة والمتعددة القيم المعرفة على R . وليكن D_i إسقاط D على R_i وتضم مجموعة الارتباطات التابعة من D^+ والحاوية فقط لوصفات R_i . ولجميع الارتباطات المتعددة القيم من الشكل R_i حيث $X \rightarrow Y$ حيث X من R_i و $X \rightarrow Y$ من D^+ ، نقول إن التجزئة محافظة على الارتباطات إذا كان (لكل مجموعة علاقات $r_1(R_1), r_2(R_2), \dots, r_n(R_n)$ حيث r_i محقة لـ D_i) يوجد علاقة $r(R)$ تحقق D و بحيث r_i إسقاط r على R_i لجميع i .

ملاحظة : إن تجزئة مخطط علاقتي إلى مجموعة مخططات علاقتية من الشكل النظامي الرابع يمكن أن تكون غير محافظة على الارتباطات.

3 - التقييس باستخدام الارتباط الدمجي

يقيد الارتباط الدمجي مجموعة العلاقات المجزئة لمخطط علاقتي R بكونها تجزئة محافظة على المعطيات، فنقول إن $R_1, R_2, R_3, \dots, R_n$ التي هي تجزئة للمخطط العلاقتي R أي

$$R = R_1 \cup R_2 \cup R_3 \cup \dots \cup R_n$$

تحقق ارتباطا دمجيا ونرمز إليه بـ $(R_1, R_2, \dots, R_n)^*$

ملاحظة :

يوجد ارتباط دمجي بديهي إذا كانت إحدى العلاقات $R_i = R$.

الارتباط الدمجي $(R1, R2)^*$ يكافئ الارتباط المتعدد القيم $R2 \rightarrow R2 \rightarrow R1 ?$ وبالمماثلة فإن الارتباط المتعدد القيم $Y \rightarrow X$ مكافئ للارتباط الدمجي $.(X \cup (R-Y), X \cup Y)$

3-1- الشكل النظامي (PJNF (Project-Join Normal Form

نقول إن المخطط العلاقتي R المعروف عليه مجموعة الارتباطات التابعة والمتعددة القيم D وارتباطات دمجية، هو من الشكل PJNF إذا كانت جميع الارتباطات الدمجية في D^+ والتي هي من الشكل $(R1, R2, \dots, Rn)^*$ حيث Ri جزء من R و $R = R1 \cup R2$ تحقق على الأقل أحد الشرطين التاليين :

$(R1, R2, \dots, Rn)^*$ ارتباط دمجي بديهي.

كل علاقة Ri هي مفتاح رئيسي لـ R .

وكما أن كل ارتباط تابعي متعدد القيم هو ارتباط دمجي، فإن كل مخطط علاقتي من النوع PJNF هو من النوع 4NF.

مثال :

ليكن لدينا المخطط العلاقتي :

Loan-info-schema = (branch-name, customer-name, loan-number, amount).

حيث بفرض أن القرض يمكن أن يكون لأكثر من زبون، ويشارك في إعطائه أكثر من فرع وله كمية معينة؛ إن هذه الارتباطات مستقل بعضها عن بعض، ذلك حيث أنه لدينا ارتباط دمجي بين العلاقات التالية :

*((loan-number,branch-name),(loan-number,customer-name),
(loan-number, amount))

المخطط العلاقتي Loan-info-schema هو ليس من الشكل النظامي PJNF مع الارتباطات التابعة الحاوية للارتباط الدمجي السابق. ولوضعه بالشكل النظامي PJNF يجب تجزئته إلى المخططات العلاقتية الثلاثة التالية :

(loan-number, branch-name)
(loan-number, customer-name)
(loan-number, amount)

4- الشكل النظامي لمجال المفتاح

Domain Key Normal Form (DKNF)

تعريف المجال : مجال تعريف واصف A هو dom يعني أن جميع القيم الذي يأخذها هذا الوصف في جميع الحدوديات هي قيمة من مجموعة القيم الموجودة في المجال dom. تعريف المفتاح : نقول إن مجموعة الواصفات K هي مفتاح للعلاقة R، ويرمز إليه بـ Key(K)، إذا كان مفتاحاً رئيسياً للمخطط العلاقتي $R(K \rightarrow R)$. تكون جميع تعاريف المفاتيح ارتباطات تابعة بين المفتاح والعلاقة وبالطبع العكس ليس بالضرورة صحيحاً. شرط تكامل عام : الشرط العام هو عبارة عن قضية يجب تحققها على علاقة.

ليكن لدينا D مجموعة من شروط المجالات ، وليكن K مجموعة من المفاتيح على المخطط العلاقتي R. لنرمز بـ G إلى مجموعة شروط التكامل العامة على R. نقول إن R هو من الشكل النظامي مجال مفتاح إذا كان اجتماع $D \cup K$ ينتج منطقياً G.

مثال :

ليكن لدينا الشرط التالي على الحسابات المصرفية : جميع الحسابات المصرفية التي أرقامها المصرفية تبدأ بالرقم 9 هي حسابات خاصة بفائدة عالية مع رصيد لا يقل عن 2500.

الشرط العام هو : " إذا كان الرقم الأول من t[account-number] هو الرقم " 9 " فإن $t[balance] \geq 2500$.

تصميم المخطط من النوع DKNF هو :

Regular-acct-schema=(branch-name, account-number, balance)

Special-acct-schema=(branch-name, account-number, balance)

شرط المجال للمخطط Special-acct-schema هو :

- أرقام الحسابات تبدأ ب 9.
- الرصيد أكبر من 2500.

الفصل السابع

المخطط الداخلي لقاعدة المعطيات

مقدمة

المستوى الفيزيائي (الداخلي) Physical level هو المستوى الأدنى في تجريد المعطيات، ويصف الطريقة الفعلية لتخزين المعطيات. يتعلق المستوى الداخلي ببنية التخزين التي ستحتوي المعطيات فعلياً، ويسمح بوصف المعطيات حسب الطريقة المتبعة في التخزين. غالباً لا يحتاج القارئون على إدارة قواعد المعطيات إلى التدخل على هذا المستوى، ويتركون هذه المهمة لنظام إدارة قواعد المعطيات الذي يقوم بترجمة النموذج المنطقي إلى نموذج فيزيائي مكافئ. لكن فهم المبادئ المتبعة في تخزين وإدارة المعطيات على المستوى الفيزيائي يصبح ضرورياً عندما نحتاج لإجراء عمليات تخص مدير قاعدة المعطيات مثل النسخ الاحتياطي، وإصلاح الأعطال، وتحسين أداء النظام، وغيرها من المهام التي تستدعي فهماً دقيقاً لبنى التخزين الفعلية المعتمدة في نظام إدارة قواعد المعطيات. يتضمن هذا الفصل تذكراً لبنى تخزين المعطيات التي درسها الطالب في مادة الخوارزميات وبنى المعطيات.

1- تعاريف

التسجيلات: تُخزن المعطيات الفعلية في مجموعة من التسجيلات، كل منها مؤلفة من حقل (Field) أو أكثر، أنماط قيم هذه الحقول هي الأنماط الأساسية كالأعداد الصحيحة

أو الأعداد الحقيقية أو سلاسل المحارف ذات الحجم الثابت. يضاف إلى هذه الأنماط الأساسية نمط المؤشر (Pointer) الذي يفيد في ربط التسجيلات.

الملف : هو مجموعة من التسجيلات لها نفس البنية. يُنجز الوصول إلى عناصر ملف بطرق مختلفة، وقد لا تكون تسجيلات الملف مخزنة بنفس التتابع.

الكتلة Block : هي وحدة التعامل الأصغر بين الذاكرة الرئيسية والذاكرة الثانوية، بحيث تحوي عملية نقل المعطيات بين الذاكرة الرئيسية والذاكرة الثانوية عدداً صحيحاً من كتل كاملة. وهذا الشرط مطبق سواء في النظم التي تدعم الذاكرة الوهمية التي تقسم إلى مجموعة كتل من ثمانيات متتابعة، أو في النظم التي لا تدعم الذاكرة الوهمية حيث يمكن أن تكون الكتلة عبارة عن قطاعاً (Sector) في مسار. يجري عادة تخزين عدد من التسجيلات على كتلة واحدة، ويجري تجميع التسجيلات في الكتل بطريقة تختصر زمن الوصول إلى المعطيات من خلال جعل عدد الكتل التي يجري تبادلها بين الذاكرة الرئيسية والذاكرة الثانوية أقل ما يمكن.

المؤشرات : المؤشر إلى تسجيلة r هو معطى كافٍ لتحديد موقع r ، و تختلف طبيعة المؤشر باختلاف بنى المعطيات المستخدمة لتخزين التسجيلات.

الوصول الكتلي : هو وحدة الكلفة للعمليات على الكتل الفيزيائية، وهي إما القراءة أو الكتابة في كتلة واحدة. يقوم نظام التشغيل أو نظام إدارة قواعد البيانات أحياناً بتخزين نسخ عن الكتل في الذاكرة الرئيسية مادام لديه متسع لذلك مما يوفر زمن القراءة والكتابة في القرص.

2- تنظيم التسجيلات

تتعامل قواعد المعطيات في المستوى الفيزيائي مع نوعين من التسجيلات :

- تسجيلات ذات طول ثابت : تحوي عدداً محدداً من الحقول لكل منها طول ثابت. يجري ترتيب هذه الحقول بطريقة يمكننا من الوصول إلى قيم هذه الحقول. يبدأ كل حقل عند عدد محدد من الثمانيات نسميه بالانزياح (Offset) اعتباراً من بداية التسجيلة. تُتيح هذه الانزياحات الوصول إلى جميع حقول التسجيلة إذا كانت بدايتها معروفة. وتُستخدم ثمانيات أخرى (ليست مخصصة لحقول المعطيات) في كل تسجيلة لتخزين خواص التسجيلة، وحجمها، وحالتها.
- تسجيلات ذات حجم متغير : تحوي حقولاً ذات أحجام متغيرة تستدعي اتباع استراتيجيات مختلفة لتمثيلها وإدارتها.

3- تمثيل الكتل

تتضمن الكتلة عدداً صحيحاً من التسجيلات. و نحتاج في الكتلة إلى أحجام إضافية لتخزين بعض المعلومات عن الكتلة مثل مؤشرات إلى مواقع ثابتة لربط الكتل إلى سلاسل الكتل. إذا كانت الكتلة تحوي تسجيلات ذات أحجام ثابتة، عندئذٍ يمكن تقسيم الكتلة إلى أحجام عديدة بحيث يشغل كل حجم منها تسجيلة واحدة مع ترك بعض الحقول الخاصة في الكتلة كالمؤشرات إلى الكتل الأخرى. إذا أردنا تخزين تسجيلات ذات أحجام متغيرة في كتلة، فإننا نحجز مجلد (Directory) في بداية الكتلة ليحتوي مؤشرات (أو جدول مؤشرات) إلى جميع بدايات التسجيلات في الكتلة.

4- الفهارس الأساسية

تتطلب نظم إدارة قواعد البيانات أو نظم إدارة الملفات إجراء العمليات الأساسية التالية:

- البحث (Lookup) : إيجاد التسجيلية (أو التسجيلات) التي تحوي في الحقول المخصصة لمفتاحها قيمة معطاة.
- الإضافة (Insertion) : إضافة تسجيلية إلى ملف حيث نفترض أن التسجيلية التي ستضاف غير موجودة سلفاً في الملف، أو أنه لا يعنينا وجود تسجيلية مطابقة أو لا.
- الحذف (Deletion) : حذف تسجيلية من ملف حيث نفترض هنا أننا لا نعرف سلفاً: أهى في الملف أم لا، لذلك فهذه العملية تتضمن حتماً عملية بحث.
- التعديل (Modification) : تغيير بعض القيم في حقل واحد أو أكثر من حقول تسجيلية، وهذه العملية تتطلب أيضاً إيجاد التسجيلية التي نريد تعديلها.

يعتمد الفهرس الأساسي (Primary Index) بشكل عام على مفتاح الملف، كما أن موقع التسجيلية في الملف يعتمد أيضاً على مفتاحها. لذلك من المهم إيجاد تسجيلية ما "بسرعة" عند إعطاء قيمة مفتاحها. كما أنه من المفيد مناقشة حالة استخدام حقل أو أكثر كمفتاح للتسجيلات التي لا تحدد تحديداً وحيداً هذه التسجيلات، لأن تطابق المفتاح بين عدة تسجيلات قد يؤدي إلى إطالة زمن البحث عن تسجيلية.

4-1- الكومة

تُعتبر الكومة (Heap) البنية أبسط بنية حيث توضع التسجيلات بلا أي ترتيب خاص في الكتل، والتي بدورها لا تخضع لترتيب معين. سنفترض فقط أن المؤشرات إلى جميع

كتل الملف مخزنة في الذاكرة المركزية. وفي حالة وجود عدد كبير من الكتل بحيث يصبح من المتعذر تخزين جميع المؤشرات إليها في الذاكرة المركزية، عندئذ يمكن تخزين هذه المؤشرات في كتل على الذاكرة الثانوية حيث تُسترجع عند الحاجة.

4-2- ملفات التقطيع

تعتمد الفكرة الأساسية للملفات التقطيع (Hashed Files) على توزيع التسجيلات على رزم (Buckets) حسب قيم مفاتيحها. يوجد تابع تقطيع (Hash Function) من أجل كل ملف مخزن بهذه الطريقة، يأخذ قيمة المفتاح ويحولها إلى عدد صحيح من المجال $[0..B-1]$ حيث B هو عدد رزم الملف.

تحتوي كل رزمة عدداً صغيراً من الكتل، حيث تنظم هذه الكتل ضمن الرزمة الواحدة على شكل كومة. نفترض إذن وجود جدول من المؤشرات مفهرس من 0 إلى B-1 ونسميه مجلد الرزم. نُعتبر كل خانة من هذا الجدول مؤشراً إلى بداية سلسلة خطية، منتهية بقيمة Null، كل عنصر منها هو كتلة ونسمي هذه الخانة بترويسة الرزمة.

4-3 الملفات المفهرسة

وتُسمى أيضاً بملفات "ايسام" (ISAM: Indexed Sequential Access Method)، التي تفترض أن المفاتيح المستعملة في التسجيلات وحيدة (Unique keys) ومن ثم تتطلب أيضاً هذه الطريقة فرز المفاتيح تصاعدياً.

نستطيع مبدئياً مقارنة قيم المفاتيح مهما كان نمطها ومن ثم فرزها. يمكن تعريف ترتيب على هذه القيم سواء أكانت أعداداً صحيحة أم حقيقية أم سلاسل محارف.

يمكن أن نستفيد من الترتيب المعرف على قيم مفاتيح التسجيلات للوصول إلى أي تسجيلية مفتاحها معروف "بسرعة".

لزيادة سرعة الوصول إلى الملف المرتب، الذي نسميه عادة بالملف الأساسي (Main file) نبني ملفاً ثانياً، نسميه بالفهرس الأجوف (Sparse Index) أو ملف الفهرس أو فهرس ايسام، الذي يحتوي على التسجيلات: (قيمة المفتاح، عنوان الكتلة).

يوجد لكل كتلة B من الملف الأساسي تسجيلية (v, B) في ملف الفهرس حيث القيمة v أصغر من أي قيمة موجودة في B وأكبر من أي قيمة موجودة في الكتلة التي تسبق B . الحقل الأول من التسجيلية (v, B) هو مفتاح أيضاً لملف الفهرس ويبقى هذا الملف مفروزاً حسب قيمة v .

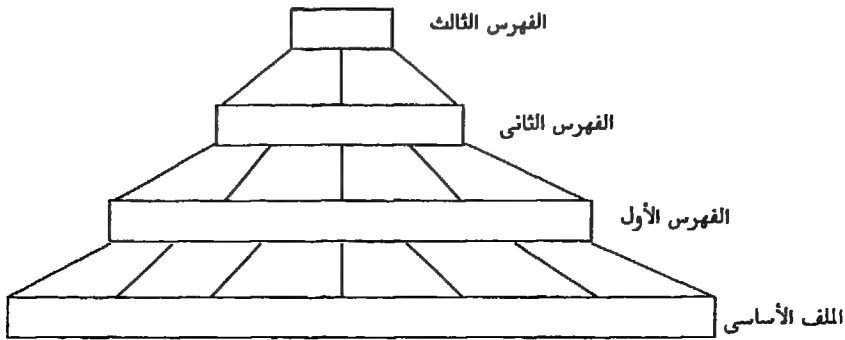
5- الفهارس المتعددة المستويات

تعتمد الفهارس متعددة المستويات مبدأ فهرسة الفهرس نفسه، ومن ثم فهرسة الفهرس الناتج، وهكذا حتى نستطيع وضع الفهرس في كتلة واحدة.

يُرتب الملف الأساسي حسب قيم المفاتيح. يحتوي الفهرس في المستوى الأول التسجيلات (v, b) حيث b مؤشر إلى الكتلة B من الملف الأساسي و v أصغر قيمة مفتاح في B . من الطبيعي أن هذه التسجيلات مرتبة حسب قيمة v . يحتوي الفهرس في المستوى الثاني أيضاً تسجيلات من الشكل (v, b) حيث b مؤشر إلى الكتلة B من الفهرس الأول و v أصغر قيمة مفتاح في B ، وهكذا.

نستطيع بهذه الفهرسة المتعددة المستويات (Multi Level Indexing) زيادة فعالية عمليات البحث والإضافة والتعديل والحذف أكثر بكثير من الفهرسة السابقة الأحادية المستوى.

تُعتبر أشجار Bayer (أو اختصاراً الشجرات B)، من أشهر بنى المعطيات التي تساعد في بناء فهارس متعددة المستويات. سندرس في هذا الفصل تأثير الفهرسة المتعددة المستويات على زيادة فعالية العمليات على المعطيات المخزنة في الذاكر الثانوية.



الشكل (1): فهرسة متعددة المستويات

تحتفظ البنية الخاصة المعروضة في الشكل (1) بالملف الأساسي في أوراق شجرة Bayer نفسها.

تُعرف أشجار Bayer عادةً لاستخدام استراتيجيات خاصة من أجل عمليات الحذف والإضافة التي تضمن بقاء جميع العقد (عدا الجذر) نصف ممثلة على الأقل.

الفصل الثامن

للمناقلات

1- تعاريف

- للمناقلة هي مجموعة من التعليمات البرمجية يتم تنفيذها سويةً (بشكل متكامل)، يمكن لهذه التعليمات أن تقوم بتعديل أو الوصول إلى مجموعة من المعطيات، ويجري بواسطتها نقل قاعدة المعطيات من حالة صحيحة إلى حالة صحيحة أخرى.
- يجب العناية عند دراسة مفهوم المناقلة بالأمور الرئيسية التالية :
- معالجة الأعطال المختلفة التي يمكن أن تحصل أثناء عمل نظم إدارة قواعد المعطيات مثل أعطال التجهيزات وأعطال النظم البرمجية ،
 - التنفيذ المتزامن لمجموعة من المناقلات
- لحفظ تكامل المعطيات، يجب أن يؤمن نظام قواعد المعطيات ما يلي :
- الكتلية : يتم تنفيذ جميع العمليات المتضمنة ضمن المناقلة بشكل متكامل وينعكس ذلك على قاعدة المعطيات أو لا يتم تنفيذ أيًا من هذه العمليات.
 - الملاءمة : يحافظ تنفيذ مناقلة على إبقاء قاعدة المعطيات صحيحة أي أن المعلومات تبقى مترابطة ومحافظة على شروط التكامل المعرفة عليها.
 - العزل : يمكن أن يسمح النظام بتنفيذ عدة مناقلات تنفيذاً متزامناً، وبمعزل عن بعضها. ويجب أن تكون النتائج الوسيطة أثناء تنفيذ المناقلة مخفية عن المناقلات الأخرى المنفذة على التوازي. (ذلك أنه من أجل كل زوج من المناقلات T_i, T_j

يبدو T_i أن T_j يبدأ التنفيذ بعد نهاية تنفيذه أو أنه يُنهي تنفيذه قبل بداية تنفيذه).

- الاستمرارية : بعد أن ينتهي تنفيذ المناقلة بنجاح فالتغيرات الناتجة على قاعدة المعطيات تصبح جزءاً من القاعدة حتى ولو حدث عطل في النظام .
مثال :

لتكن لدينا المناقلة التي تسمح بتحويل مبلغ 500 ل.س من حساب A إلى حساب B التالية :

1. Read (A)
2. $A := A - 500$
3. Write (A)
4. Read (B)
5. $B := B + 500$
6. Write (B)

يجب أن يحقق التنفيذ الخواص التالية :

الملاءمة : لا يتغير مجموع الحسابين المصرفين A, B بعد تنفيذ المناقلة.

الكتلية : يتم تنفيذ جميع التعليمات المتضمنة في المناقلة ككتلة واحدة فإذا حصل خطأ أثناء التنفيذ في مرحلة ما، لا تتفعل العمليات التي أجريت قبل هذه المرحلة.

الاستمرارية : التعديلات الناتجة من تنفيذ المناقلة على قاعدة المعطيات تبقى حتى ولو تعرضت القاعدة إلى أعطال.

العزل : في حال سُمح بتنفيذ مناقلة أخرى، تحتاج للوصول إلى قاعدة المعطيات المعدلة جزئياً، بين الخطوتين 3 و 6. فإن هذه المناقلة ستتعامل مع قاعدة معطيات غير ملائمة (غير صحيحة) حيث سيكون مجموع $A+B$ أقل من المجموع الحقيقي.

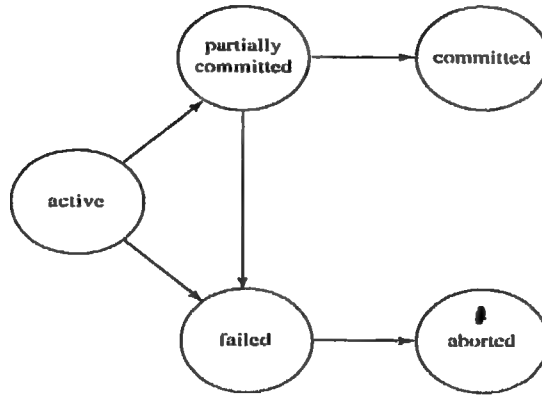
يمكن تحقيق هذه الخواص بسهولة من خلال تنفيذ المناقلات تسلسلياً، أي الواحدة تلو الأخرى، بينما تنفيذ المناقلات بشكل متزامن لها فوائد متعددة كما سنرى.

2- الحالات المختلفة للمناقلة

تكون المناقلة إحدى الحالات التالية :

- فعالة (Active) وهي الحالة البدائية، وتبقى المناقلة في هذه الحالة أثناء التنفيذ.
- منفذة جزئياً (Partially committed) بعد تنفيذ آخر تعليمة.
- فاشلة (Failed) عند اكتشاف عدم إمكانية متابعة التنفيذ بشكل صحيح.
- متراجعة (Rolled Back) بعد أن يتم التراجع عن تنفيذ المناقلة وتقوم قاعدة المعطيات بإعادة تخزين حالتها السابقة لتنفيذ المناقلة. ولدينا حالتان للمتابعة بعد أن يتم التراجع عن تنفيذ المناقلة، وهما :
 - إعادة تنفيذ المناقلة فقط في حال عدم وجود خطأ منطقي
 - إيقاف تنفيذ المناقلة.
- مثبتة (Committed) بعد إتمام التنفيذ بنجاح.

يبين الشكل التالي دورة حياة المناقلة :



تقوم نظم قواعد المعطيات (الجزء الخاص بإدارة الاسترجاع) بتوفير الأدوات اللازمة لضمان تنفيذ المناقلة بشكل متكامل (الكتلية) واستمرار تأثيرها على القاعدة، ويجري تحقيق ذلك باعتماد الخطوات التالية :

- التأكد من وجود مناقلة واحدة فعالة بوقت واحد، واستخدام مؤشر يؤشر دوماً إلى نسخة قاعدة المعطيات الصحيحة، وإجراء التعديلات المطلوبة ضمن المناقلة على نسخة من القاعدة، وعند إنهاء تنفيذ جميع التعليمات المطلوبة بنجاح يتم نقل المؤشر إلى النسخة الصحيحة المعدلة. في حال حدوث خطأ أثناء التنفيذ فيمكن استخدام قاعدة المعطيات المؤشر إليها وحذف النسخة الاحتياطية غير الصحيحة. يتطلب نجاح تلك الطريقة عدم حصول أخطاء في الأقراص، وهي غير مناسبة لقواعد المعطيات الضخمة لأن تنفيذ مناقلة واحدة يتطلب نسخ لقاعدة المعطيات بأكملها.

- في حال السماح بتنفيذ مجموعة من الناقلات بشكل متزامن، فإن الأمر يتطلب وجود تقنيات ترتيب زمني للتنفيذ للتحكم في تنفيذ تعليمات الناقلات المتزامنة بحيث تبقى قاعدة المعطيات صحيحة.
- الترتيب الزمني للتنفيذ Schedules هو تسلسل يبين الترتيب المنطقي الذي يتم فيه تنفيذ تعليمات الناقلات المتزامنة. ويجب مراعاة ما يلي :
 - احتواء الترتيب الزمني للتنفيذ على جميع التعليمات الموجودة في جميع الناقلات،
 - المحافظة على الترتيب التي تظهر فيه هذه التعليمات في كل مناقلة على حدى.
- مثال : لتكن لدينا المناقلة T1 والتي تقوم بتحويل \$50 من الحساب A إلى الحساب B، وT2 تقوم بتحويل 10% من رصيد الحساب A إلى الحساب B.
- نقول بأن الترتيب الزمني التالي لتنفيذ الناقلتين تنفيذ متداخل والذي هو عبارة عن تنفيذ تسلسلي لـ T1 ومن ثم T2.

T1	T2
Read (A);	
A := A-50	
Write (A);	
Read (B);	
B := B +50;	
Write (B);	
	Read (A);
	Temp := A*0.1
	A := A- temp
	Write (A);
	Read (B);
	B := B +temp;
	Write (B)

نلاحظ أن نتيجة تنفيذ البرنامجين السابقين بالترتيب الزمني السابق يُحافظ على بقاء مجموع A و B ثابتاً. بينما نتيجة التنفيذ بالترتيب الزمني التالي لا يحافظ على بقاء مجموع A و B ثابتاً.

T1	T2
Read (A);	
A := A-50	
	Read (A);
	Temp := A*0.1
	A := A- temp
	Write (A);
	Read (B);
Write (A);	
Read (B);	
B := B +50;	
Write (B);	
	B := B +temp;
	Write (B);

القواعد الأساسية :

- يجب أن يحافظ تنفيذ أي مناقلة على تجانس قاعدة المعطيات.
- يجب أن يحافظ تنفيذ مجموعة مناقلات متسلسلة على تجانس قاعدة المعطيات.
- نقول عن مخطط تنفيذ متداخل أنه قابل للترتيب serializable إذا كان قابلاً للتحويل إلى تنفيذ بترتيب زمني معين للمناقلات. أي أن أي تنفيذ للعمليات الأساسية للبرامج (المناقلات) T_i يحترم ترتيب العمليات الواردة ضمن البرامج T_1, T_2, \dots, T_n .

3- الأقفال (Locks)

هي وسيلة التحكم الأساسية بعناصر المعلومات. ويُشكل مدير الأقفال Lock Manager أحد المكونات الأساسية في نظام إدارة قواعد المعطيات، فهو يهتم بمراقبة الوصول إلى عناصر المعلومات ويُنظم عمليات القراءة والكتابة بحيث يضمن صحة المعلومات وعدم ضياعها.

3-1- الأقفال وإدارة الوصول المتزامن إلى عناصر المعلومات

لتكن لدينا منافقتان تنفذان برنامجاً حسب الترتيب الزمني التالي :

T1	T2
Read (A);	Read (A);
A := A+1	A := A+1
	Write (A)
Write (A)	

إن تنفيذ المنافقتين بالترتيب الزمني السابق سوف يُعطي نتيجة مغلوطة لقيمة المتحول A، والحل البديهي هو أن تقوم كل منافقة بحجز عناصر المعلومات التي تريد قراءتها قبل إجراء عملية القراءة وتقوم بتحريرها بعد إجراء عملية الكتابة.

يُصبح البرنامج السابق كما يلي :

```

Lock (A);
Read (A);
A := A+1;
Write(A);
UnLock(A);
    
```

3-2- بعض مشاكل الأقفال

إن استخدام مفهوم الأقفال يحل مشكلة المشاركة في استخدام عناصر المعلومات، ولكن من الممكن أن ينتج عنه مشاكل أخرى متعددة منها :

- الانتظار اللانهائي Live Lock : حيث من الممكن أن يطلب T1 قفلاً على عنصر مثل A المحجوز لتنفيذ المناقلة T2، وعندما يُنهى T2 عمله يقوم بتحرير A الذي يقوم النظام بحجزه للمناقلة T3 التي طلبت القفل عليه أثناء تنفيذ T2، وهكذا يمكن أن يُمنح القفل بعد ذلك لـ T4، ... نلاحظ أن T1 ينتظر بشكل لا نهائي للحصول على قفل لا يحصل عليه لأن النظام يُفضّل في كل مرة مناقلةً أخرى.

- العرقلة المتبادلة Dead Lock : وذلك عند انتظار مناقلة x عنصر معلومات مقفول من مناقلة Y، التي بدورها تنتظر عنصر معلومات مقفول من المناقلة X.

مثال :

لتكن لدينا الناقلتان التاليتان :

T1 : Lock A; Lock B;; UnLock A; UnLock B;

T2 : Lock B; Lock A;; UnLock B; UnLock A;

وبفرض أن T1, T2 يبدأان التنفيذ في وقت واحد، نجد أن T1, T2 ينتظران بشكل لا نهائي.

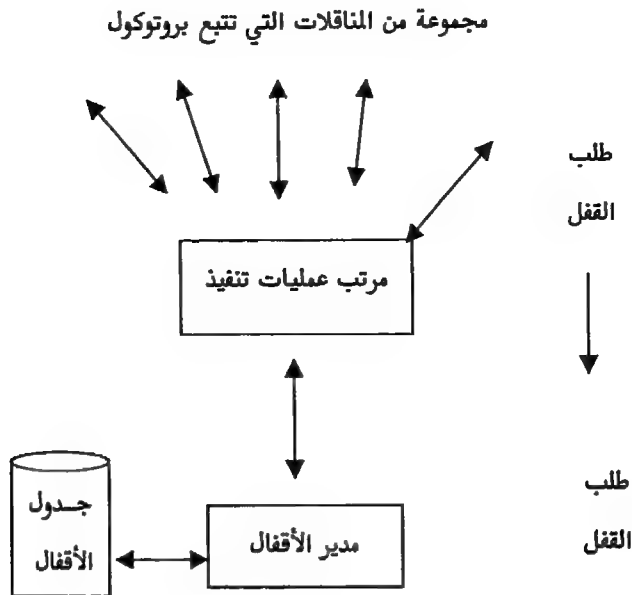
توجد عدة حلول لحل مثل هذه المسألة :

تقوم كل مناقلة بطلب كافة الأقفال التي تحتاجها على عناصر المعلومات دفعة واحدة فإما أن تحصل عليها جميعاً أو أن لا تحصل على أية منها، وإذا لم تحصل عليها فإنها تنتظر.

ترتيب عناصر المعلومات والطلب إلى كافة المنافلات أن تطلب الأقفال على العناصر التي تحتاجها وفقاً لهذا الترتيب.

4- منسق المنافلات

يبين الشكل التالي كيفية عمل منسق المنافلات الموجود ضمن نظم إدارة قواعد المعطيات :



6- نموذج مناقلة

وهو عبارة عن مجموعة من القواعد والضوابط المفروضة على المناقلات والتي تضمن طريقة معينة في التنفيذ المتداخل للمناقلات.

يمكننا النظر إلى مناقلة كمتتالية من عمليات الإقفال (حجز مصادر المعلومات) وتحرير الأقفال، وبالتالي فإن نموذج المناقلة يجب أن يحقق ما يلي :

تحرير كل عنصر جرى إقفاله

لا تقوم المناقلة بطلب قفل لعنصر تحجزه

لا تقوم المناقلة بتحرير عنصر غير محجوز لصالحها

عموماً تتبع كل عملية Lock عملية قراءة، وتسبق كل عملية UnLock عملية كتابة.

7- خوارزمية تحويل تنفيذ متداخل إلى تنفيذ متسلسل

بفرض S تنفيذ متداخل لمجموعة مناقلات T_1, T_2, \dots, T_k .

ننشئ بيان G_i عقده هي المناقلات وتُحدد الأسهم فيه بالشكل التالي :

بفرض $s = a_1, a_2, \dots, a_n$ حيث a_i تعليمة من الشكل :

$T_j : \text{UnLock } A_m$ أو $T_j : \text{Lock } A_m$

فإذا كانت $T_j : \text{UnLock } A_m$: $T_j : a_i$ ووجدت تعليمة أخرى مثل $T_s : \text{Lock } A_m = a_p$ فإننا نُنشئ سهماً من T_j إلى T_s . ومعنى ذلك أنه في أي تنفيذ متسلسل يجب أن يسبق $T_s \prec T_j$.

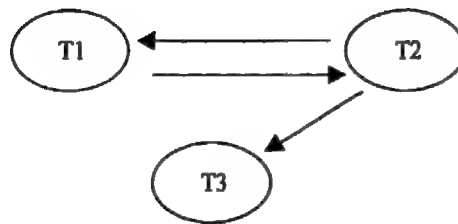
إذا وجدت حلقات في البيان الناتج فإن التنفيذ المتداخل غير قابل للتحويل إلى تنفيذ متسلسل.

مثال :

ليكن لدينا التنفيذ المتداخل التالي

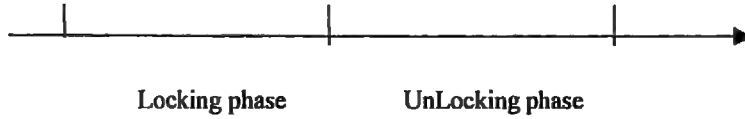
	T3	T2	T1
1			Lock A
2		Lock B	
3		Lock C	
4		UnLock B	
5			Lock B
6			UnLock A
7		Lock A	
8		UnLock C	
9		UnLock A	
10	Lock A		
11	Lock C		
12			UnLock B
13	UnLock C		
14	UnLock A		

البيان المكافئ للتنفيذ هو :



8- بروتوكول الإقفال على مرحلتين

تعريف : نقول إن مناقلة تحقق بروتوكول الإقفال على مرحلتين Two phase Locking Protocol إذا كانت كل عمليات الإقفال تسبق كل عمليات تحرير الأقفال. ويمكن برهان أن أي تنفيذ متداخل لمجموعة مناقلات تحترم هذا البروتوكول قابل للتحويل إلى تنفيذ متسلسل.



8-1- أقفال القراءة والكتابة

يجري عادة تعريف نوعين من الأقفال :

أقفال القراءة (أو الأقفال المشتركة) : عندما يتضمن برنامج مناقلة تجري عمليات قراءة فقط للعنصر A فإن هذا البرنامج يُنفذ التعليمة Rlock A ، ويمكن لمناقلات أخرى أن تحجز العنصر A للقراءة.

أقفال الكتابة Exclusive Lock عند احتواء مناقلة تعليمات تغير قيمة العنصر A فإنه يضع قفل كتابة عليه. ولا يمكن لمناقلة أخرى وضع أي نوع آخر من الأقفال على العنصر A.

يجري تحرير نوعي الأقفال بواسطة تعليمة UnLock

خوارزمية تحديد إمكانية تنسيق المنافقات :

ننشئ البيان G_i بالشكل التالي :

بفرض T_i يحجز A للقراءة ثم يحررها ، T_j يحجز A للكتابة بعد تحريرها من قبل T_i ننشئ سهماً من T_i إلى T_j .

T_i يحجز A للكتابة T_j يحجز A للقراءة بعد تحريرها من قبل T_i . ننشئ سهماً من T_i إلى T_j .

في حال احتوى البيان G_i على حلقات كان التنفيذ المتداخل للمنافقات غير قابل للتنسيق. وفي حال لم يحوي G_i على حلقات نقوم بفرز G_i فرزاً طبولوجياً.

8-2- اكتشاف العرقلة المتبادلة

يجري اكتشاف العرقلة المتبادلة عن طريق تعريف وإدارة بيان يُدعى ببيان الانتظار. تمثل عقد بيان الانتظار المنافقات ويصل سهم بين عقدتين T_i, T_j إذا كان T_i ينتظر T_j من أجل عنصر A .

تجري إدارة البيان بالشكل التالي :

عندما تطلب مناقلة T_i حجز عنصر A محجوز من قبل T_j ، نرسم سهماً صادراً من T_i إلى T_j .

عندما يحرر T_j العنصر A فإننا نحذف السهم الصادر من T_i إلى T_j .

عندما ينهي T_j تنفيذه بشكل عادي أو إجباري فإننا نحرر كافة الأسهم الواردة إليه.

عند اكتشاف وجود حلقات في بيان الانتظار فذلك يعني وجود عرقلة متبادلة ، وفي هذه الحالة يتم حذف (إجبار إنهاء) إحدى المنافقات الداخلة في الحلقة.

طريقة فحص البيان :

يمكن اعتماد إحدى الطرق التالية لفحص بيلن الانتظار :

- فحص بيان الانتظار بشكل دوري (مثلاً كل خمسة دقائق) ؛
- إعطاء زمن انتظار أعظمي لكل مناقلة فإذا تجاوز انتظارها هذا الحد الأعظمي تنفيذ تعليمة ROLLBACK ثم تقلع من جديد بعد انتظار فترة معينة.
- قبل إضافة أي سهم نتيجة تنفيذ مناقلة T_i في بيان الانتظار تجري مناقشة احتمال تشكيل حلقة نتيجة إضافة السهم. وفي حال كان ذلك محققاً يمكننا اتباع أحد المبدأين التاليين :
- اتباع مبدأ Wait-Die وإنهاء عمل المناقلة T_i بتنفيذ تعليمة RollBack.
- اتباع مبدأ Wait-Wound وإنهاء عمل إحدى المناقلات المشكلة للحلقة.

الفصل التاسع

معالجة الأعطال

تعريف

العطل هو تخريب في وسط التخزين يؤدي إلى ضياع المعلومات جزئياً أو كلياً. ومصادر الأعطال متعددة نورد منها :

- فشل تنفيذ مناقلة : هو عدم تنفيذ مناقلة نتيجة لأخطاء منطقية (لا يتم تنفيذ المناقلة لعدم تحقيق بعض الشروط الداخلية الواجبة للتنفيذ)، أو لأخطاء النظام (حيث يجب على نظام إدارة قواعد المعطيات إنهاء عمل مناقلة تؤدي إلى خطأ يخرق تكامل القاعدة).
- أعطال نظام التشغيل : والتي يمكن أن تنجم عن انقطاع التيار الكهربائي أو حدوث خطأ في نظام التشغيل ناجم عن عطل في التجهيزات أو البرمجيات. إن مثل هذا العطل يؤدي إلى إيقاف كافة التطبيقات الجارية.
- أعطال وسط التخزين : إن تخريب بعض مناطق القرص أو تعطل القرص بشكل كامل أو إجراء كتابة غير مشروعة في بعض مناطق القرص تؤدي إلى ضياع كلي أو جزئي للمعلومات.
- أعطال شاملة : كحدوث حريق يؤدي إلى ضياع النظام بأكمله.

1- الاحتياطات الأولية

يمكن استخدام نوعين من وسائط التخزين :

- وسط تخزين متبدل volatile storage وهو وسط لا يتأثر بأعطال نظام التشغيل (مثل الذاكرة الرئيسية و الذاكرة المخبئية).
- وسط غير متبدل nonvolatile storage وهو وسط يتأثر بأعطال نظام التشغيل (مثل الأقراص أو الذاكرة الثانوية والأشرطة).

ومن البديهي البحث عن وسط تخزين مستقر Stable storage مقاوم لجميع الأعطال أو استخدام مجموعة من النسخ المحفوظة على وسائط تخزين غير متبدلة متميزة. يمكن الحصول على تنفيذ مستقر للبرامج التي تعالج قاعدة المعطيات، يحميها من الأعطال، باتباع ما يلي :

- الاحتفاظ دوماً بنسخ احتياطية إضافية عن قاعدة المعطيات، ووضع هذه النسخ في مكان أمين ومحمي من الأخطار (الحريق مثلاً..). ويجري عادة الاحتفاظ بنسخ : شهرية ، أسبوعية ، يومية.

- استخدام عدة أقراص تعمل على التوازي Disk Mirroring .

يمكن أن يحصل خطأ أثناء نسخ المعطيات مما يؤدي إلى تكوين نسخ غير متماثلة، ومن أجل الحد من ذلك نقوم بإجراء عملية النسخ خارجياً بحيث يتم إجراء نسختين لكل كتلة من المعلومات بالشكل التالي :

- كتابة المعلومات على الكتلة الفيزيائية الأولى.

عند انتهاء الكتابة بشكل ناجح، نقوم بكتابة نفس المعلومات على الكتلة الفيزيائية الثانية .

تنتهي عملية النسخ الخارجي فقط عند انتهاء عملية الكتابة الثانية بنجاح. إذا احتوت إحدى النسختين على كتلة معطوبة (وجود خطأ ما)، يجري نسخها من النسخة الأخرى. وفي حال لم تحو النسختان على خطأ، ولكنهما كانتا مختلفتين، يجري نسخ النسخة الثانية على النسخة الأولى.

2- الوصول للمعطيات

لدراسة الأعطال الممكن حدوثها أثناء هذه العملية وكيفية معالجتها، سنقوم بشرح كيفية الوصول إلى المعطيات المخزنة ضمن قاعدة المعطيات.

بعض التعاريف :

الكتلة الفيزيائية Physical Block هي الكتلة الموجودة على القرص. الكتلة الدائرية هي الكتلة الموجودة بشكل مؤقت في الذاكرة المركزية.

تتم حركة الكتل بين القرص والذاكرة المركزية من خلال عمليتين :

- $input(B)$ تنقل الكتلة الفيزيائية B من القرص إلى الذاكرة المركزية.
 - $output(B)$ تنقل الكتلة الدائرية B إلى القرص، وتضع الكتلة الفيزيائية مكانها.
- لكل مناقلة T_i منطقة عمل خاصة بها تُحفظ فيها النسخ المحلية لجميع عناصر المعطيات التي تتعامل معها. تُسمى النسخة المحلية لعنصر المعطيات X بـ X_i .
- نفترض لتبسيط العمل أن كل عنصر معطيات مُخزن ضمن كتلة واحدة.

تنقل المناقلة المعطيات التي تتعامل معها بين كتل الدائري الخاصة بالنظام و منطقة العمل الخاصة بها باستخدام العمليات التالية :

• $read(X)$ والتي تقوم بوضع قيمة العنصر X في المتحول xi .

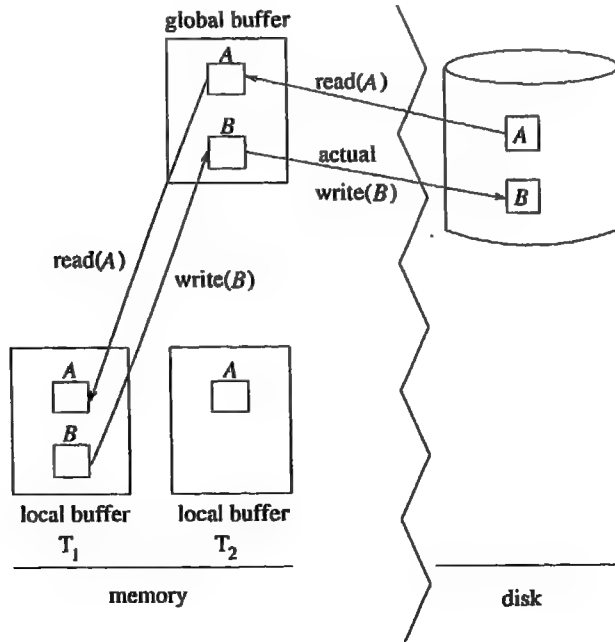
• $Write(X)$ وتسند قيمة المتحول xi إلى العنصر X .

تحتاج التعليمتان السابقتان إلى تنفيذ تعليمة $input(BX)$ ، والتي تقوم بوضع الكنلة الحاوية على العنصر X في الذاكرة (إذا لم يكن موجوداً أصلاً في الذاكرة).

تقوم المناقلة بطلب $read(X)$ عند أول وصول لـ X ؛ ومن ثم يجري استخدام نسخة محلية للمتحول المتعلق بالعنصر. تقوم المناقلة بتنفيذ تعليمة $write(X)$ بعد آخر وصول لـ X .

لا نحتاج إلى تنفيذ تعليمة $output(BX)$ فوراً بعد تنفيذ تعليمة $write(X)$.

يبين الشكل (1) التالي عملية الوصول إلى قاعدة المعطيات .



الشكل (1) : مبدأ الوصول إلى قاعدة المعطيات

3- معالجة الأعطال

لنأخذ المناقلة التي تقوم بنقل مبلغ 50\$ من الحساب A إلى الحساب B؛ هدفنا هو إما أن تتم جميع العمليات والتعديلات المطلوبة في المناقلة على قاعدة المعطيات أو أن لا يجري أي منها. بالإمكان حصول عطل ما أثناء تنفيذ المناقلة وقبل إنهاء جميع التعديلات المطلوبة على قاعدة المعطيات. فمن أجل تحقيق هدفنا والمحافظة على الكتلية للمناقلة، ننسخ المعلومات المطلوب تعديلها إلى وسط تخزين مستقر دون تعديل قاعدة المعطيات نفسها. ومن ثمّ تطبق أحد المنحيين التاليين :

• مبدأ: Log recovery الصحيفة

• مبدأ الظل shadow-paging .

ونفترض منذ البداية أن المناقلات تُنفذ بشكل متسلسل الواحدة تلو الأخرى.

مبدأ الصحيفة

الصحيفة هي ملف تسلسلي يتم الاحتفاظ به عادة على وسط تخزين مستقر مختلف عن وسط التخزين الأساسي الذي يحوي قاعدة المعطيات (عادة شريط مغناطيسي).

تحوي الصحيفة عادة تسجيلات تحدد فعالات التعديل على قاعدة المعطيات :

- عند بداية تنفيذ مناقلة يجري تسجيل التسجيلة $\langle Ti \text{ start} \rangle$ على الصحيفة.
- قبل تنفيذ تعليمة $write(X)$ ، يجري تسجيل التسجيلة $\langle Ti, X, V1, V2 \rangle$ على الصحيفة حيث $V1$ قيمة العنصر X قبل الكتابة، $V2$ القيمة التي سوف يأخذها العنصر X .
- عند انتهاء عمل المناقلة Ti (تنفيذ آخر تعليمة في المناقلة) يجري تسجيل التسجيلة $\langle Ti \text{ commit} \rangle$ على الصحيفة.

تمكننا هذه الطريقة من السيطرة على الأخطاء عن طريق تسجيل جميع التعديلات على الصحيفة وتأجيل جميع عمليات الكتابة إلى ما بعد القيام بعملية الانتهاء الجزئي للمناقلة. كما تؤكد على تنفيذ المناقلات بشكل تسلسلي . عند حدوث عطل تحتاج المناقلة إلى إعادة تنفيذ في حال جرى تسجيل كل من التعليمتين $\langle Ti \text{ start} \rangle$ و $\langle Ti \text{ commit} \rangle$ على الصحيفة.

مثال : لتكن لدينا المناقلتان $T0$ و $T1$ التاليتان ، ولنفرض أن $T0$ يجري تنفيذها قبل $T1$:

T0 :read (A)	T1: read(c)
A := A -50	C := C- 100
Write(A)	Write(C)
B := B+50	
Write (B)	

وليكن محتوى الصحيفة هو على النحو التالي في لحظات التنفيذ المختلفة

<T0 start>	<T0 start>	<T0 start>
<T0, A, 950>	<T0, A, 950>	<T0, A, 950>
<T0, B,2050>	<T0, B,2050>	<T0, B,2050>
	<T0 commit>	<T0 commit>
	<T1 start>	<T1 start>
	<T1, C, 600>	<T1, C, 600>
		<T1 commit>
(a)	(b)	(c)

بفرض حصول عطل للنظام في إحدى الحالات :

الحالة (a) لا حاجة لأي إجراء لإصلاح العطل.

الحالة (b) يجب أن يعاد تنفيذ الناقل T0 لأنه جرى تسجيل <T0 commit> في الصحيفة.

الحالة (c) يجب أن يعاد تنفيذ الناقلتين T0 و T1 بالتسلسل لأنه جرى تسجيل التسجيلتين <T0 commit> و <T1 commit> في الصحيفة.

المراجع

- [1] E.F. CODD,
A Relational Model of Data for Large Shared Data Banks,
Communications A.C.M, vol. 13, n. 6, 1970.
- [2] C.J. DATE,
An Introduction to Data Base Systems,
volumes 1 and 2, Addison Wesley Publishing Company,
third edition, 1981.
- [3] D. MAIER,
The theory of relational database,
PITMAN, 1983.
- [4] K. PARSAYE, M. CHIGNELL, S. KHOSHAFIAN, H. WONG,
Intelligent databases,
WILEY (New york), 1989.
- [5] J.D. ULLMAN,
Principles of Data Base Systems,
PITMAN (London), 1980.
- [6] A. SILBERSCHATZ, H. F. KORTH, S. SUDARSHAN
Database system concepts
Mc Graw-hill, 1998



صدر بإشراف لجنة الإنجاز

سعر المبيع للطالب (٨٠) ل.س